**北京印刷学院**

项目报告书

课程名称：

**《专业综合实训课程》**

专 业： 班 级：

**计算机科学与技术**

**2班**

学生姓名： 学 号：

**孙彬祺 /林兴博**

**224010119/117**

**刘犇**

指导教师：

时 间： 年 月 日

**22**

**12**

**2025**

目录

**[一、项目名称：基于Java EE的在线商城系统 3](#_Toc197853762)**

**[二、项目简介 3](#_Toc197853763)**

**[三、主要功能 3](#_Toc197853764)**

**[四、项目进度计划 4](#_Toc197853765)**

**[五、分工 13](#_Toc197853766)**

**[六、项目成果 13](#_Toc197853767)**

**[七、部署说明 13](#_Toc197853768)**

**[八、总结 13](#_Toc197853769)**

**[九、附录 14](#_Toc197853770)**

## 一、项目名称：基于Java EE的在线商城系统

**二、项目简介**

本项目基于 JavaEE 技术栈，开发一个功能完善的在线商城系统，包含用户管理、商品管理、订单管理及积分系统。系统采用 JSF 实现前端界面，JPA 进行数据持久化，CDI 实现依赖注入，确保模块化与可扩展性。用户可通过 Web 界面浏览商品、购买、查看订单及使用积分抵扣；管理员可额外管理商品和用户。项目使用 MySQL 数据库，通过 JNDI 配置数据源，支持高效数据管理。

**三、主要功能**

①用户管理

用户注册、登录、个人信息修改。

②商品管理

商品信息的增删改查（名称、价格、库存、图片等）；商品分类。

③订单管理

购物车功能；提交订单；订单查询、修改（支付、取消）。

④积分系统

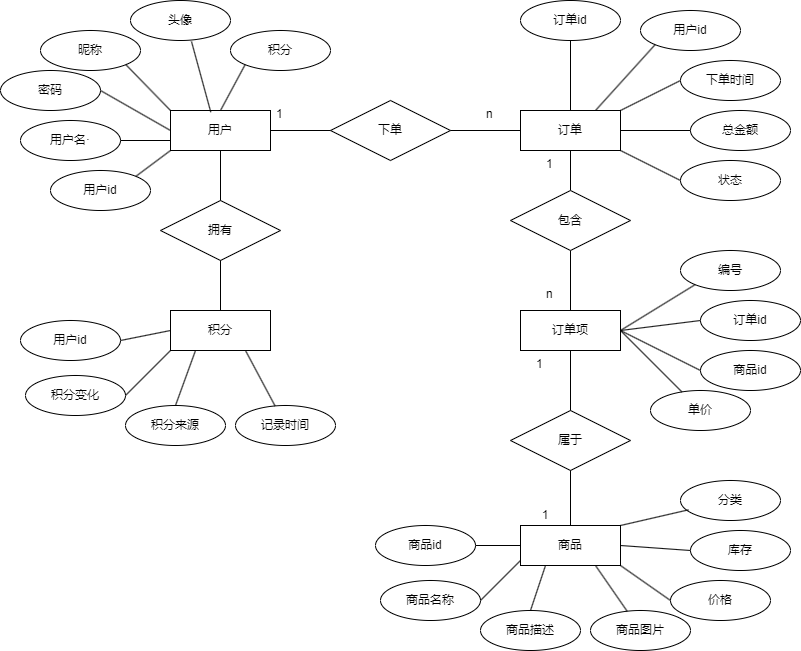
用户可通过购买商品累积积分（10元：1积分），结款时可使用积分进行（100积分：1元）比例的金额抵扣。

⑤数据存储

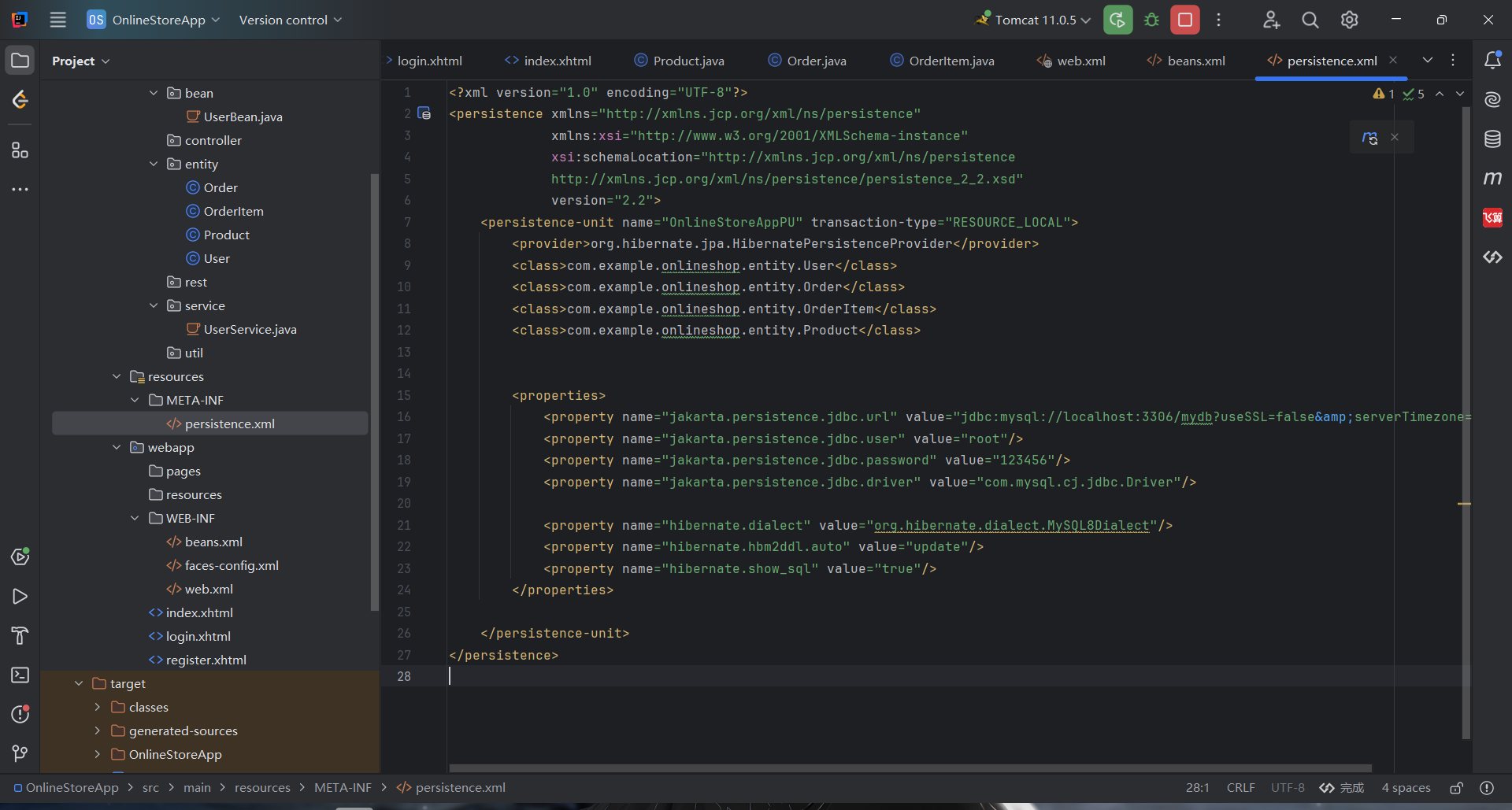
使用 JPA 进行数据库管理；通过 JNDI 进行数据源配置。

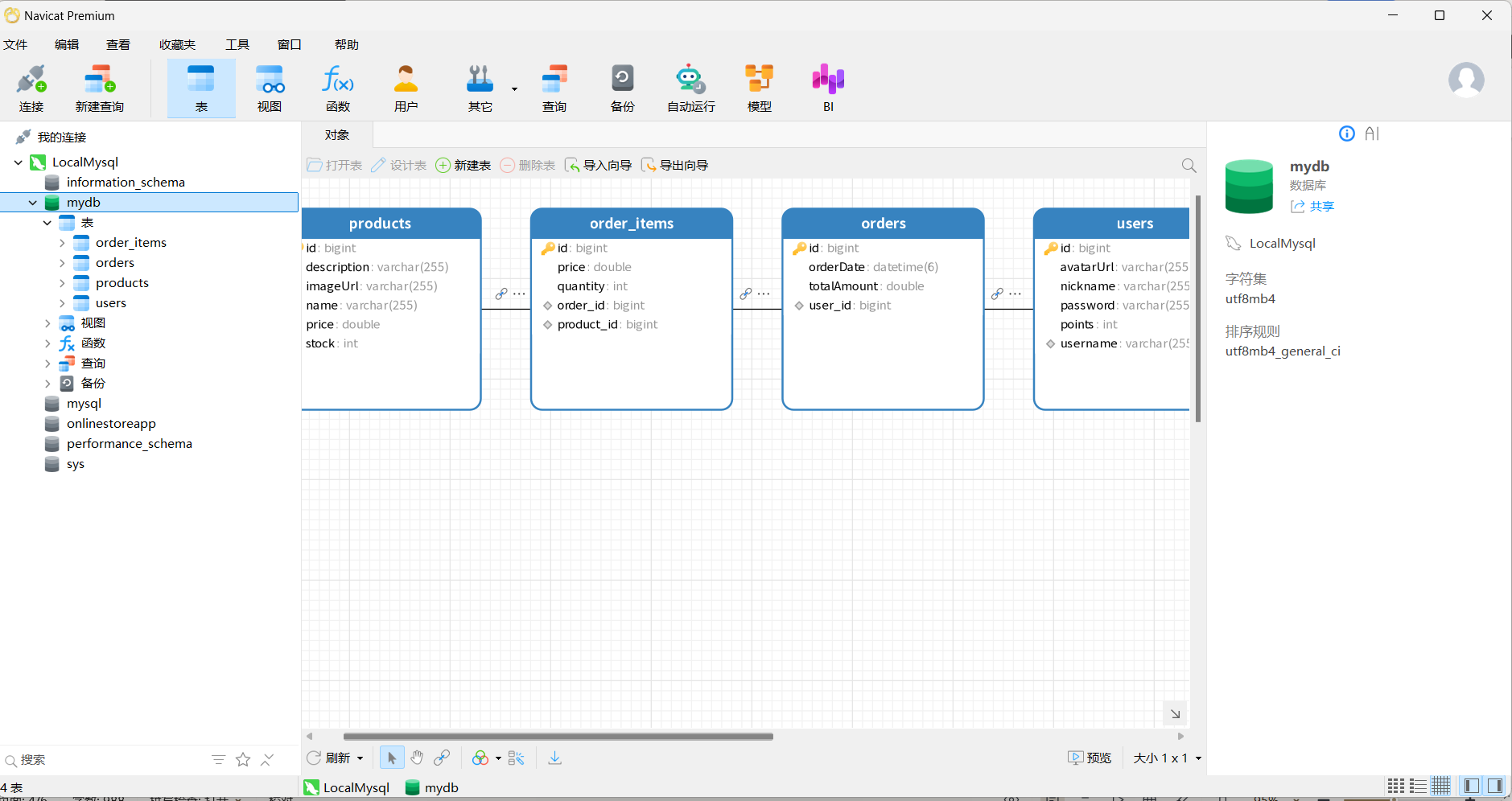
**四、项目进度成果：**

已完成项目需求分析，编写 ER 图、功能设计文档，

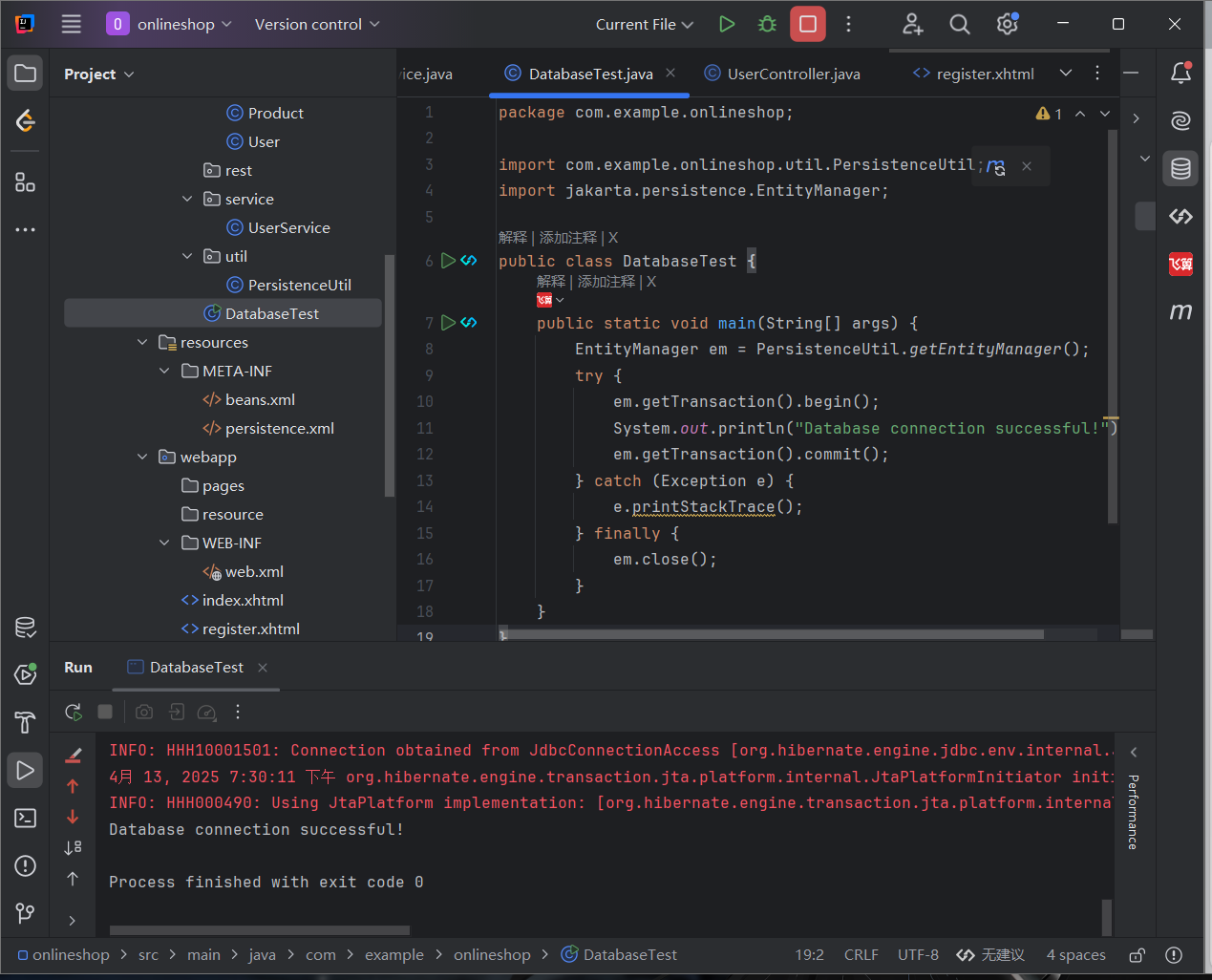


搭建 JavaEE 开发环境，创建数据库（MySQL ）

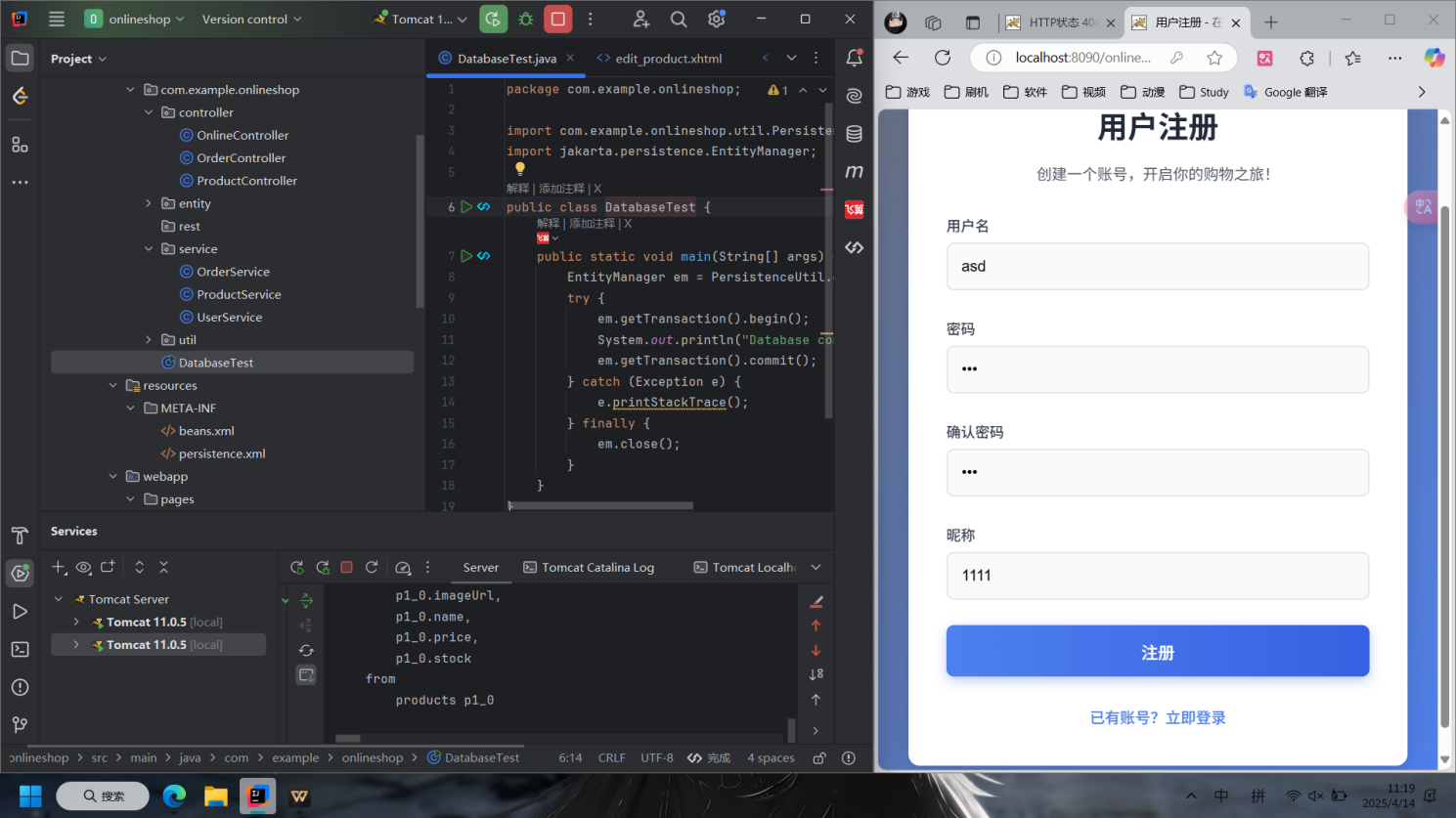


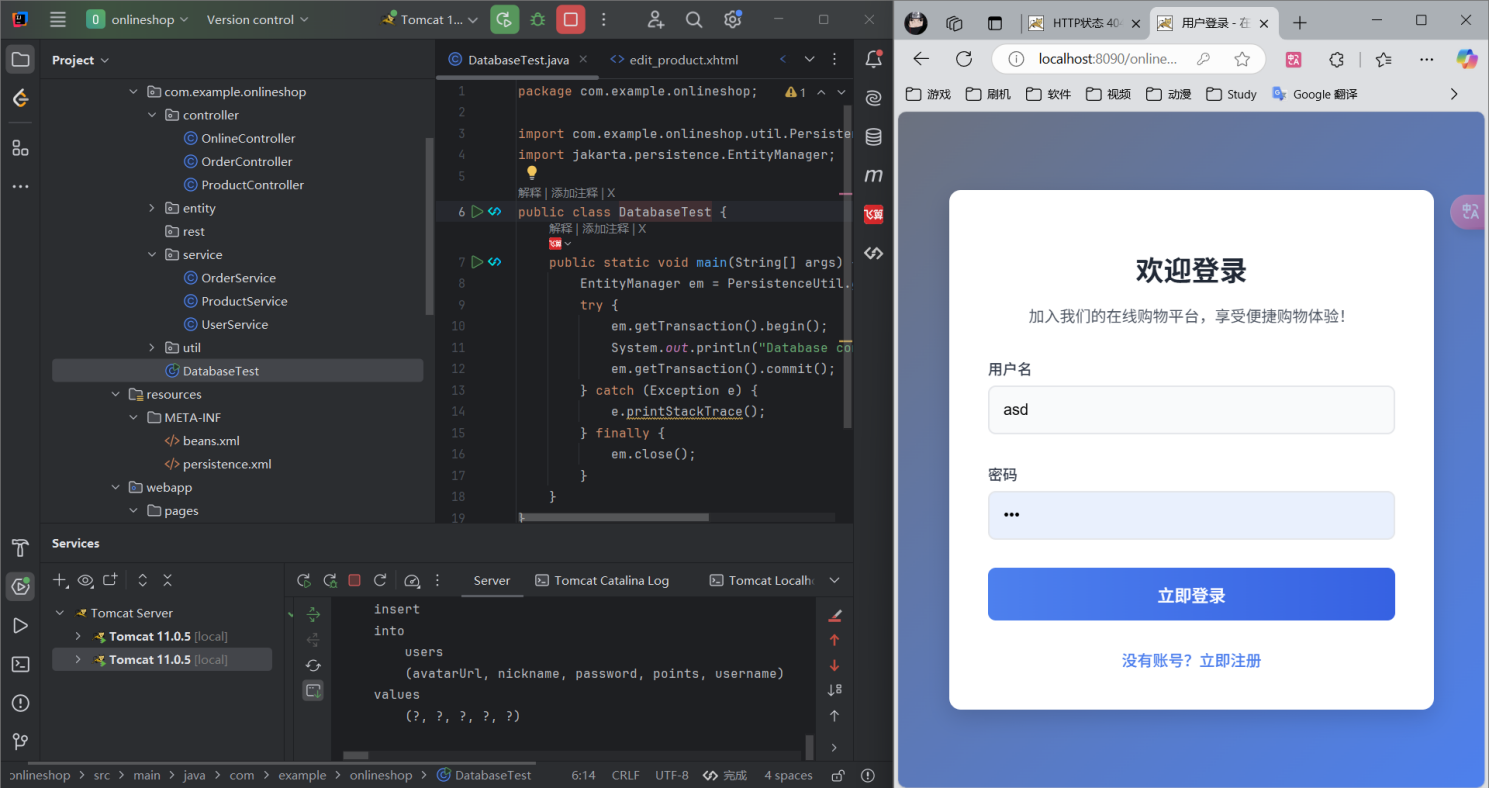
基本项目结构搭建完毕，测试并已能够实现Mysql连接

如果连接成功则出现“Database connection successful!”



实现用户管理模块（CDI + JPA），包括注册、登录、积分管理

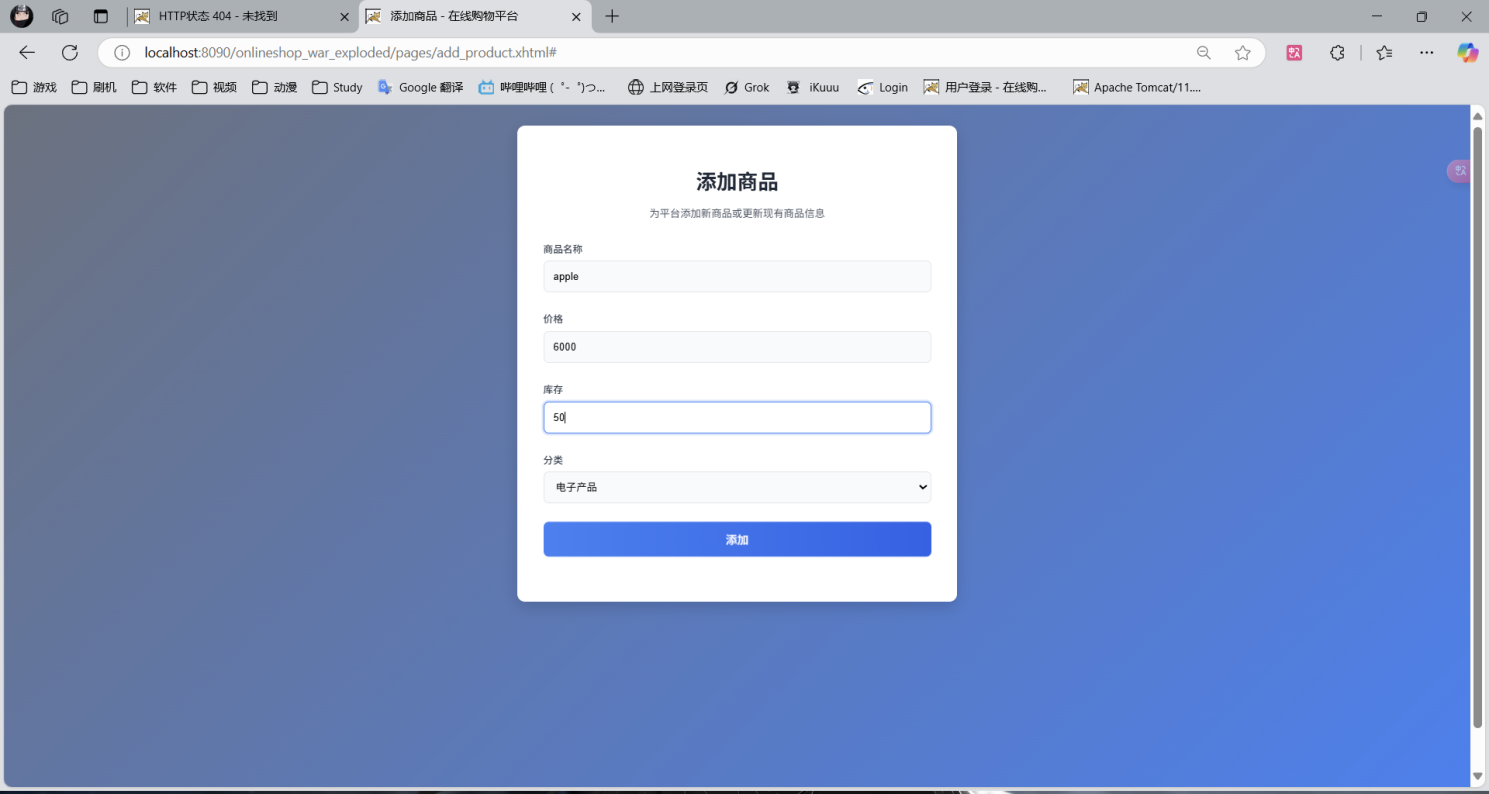




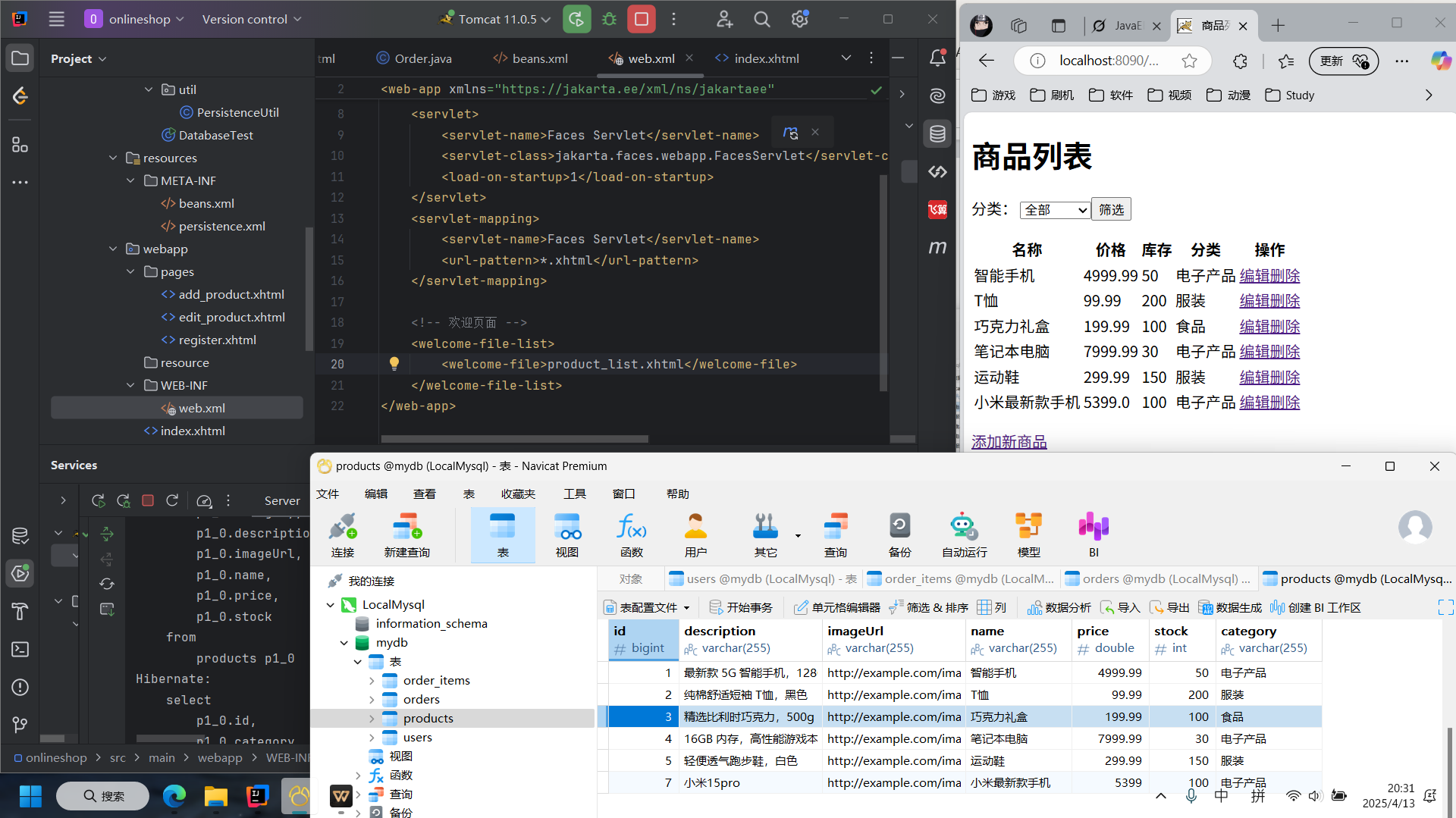
第 9 周 ：实现商品管理模块（JSF + JPA），支持商品的增删改查及分类

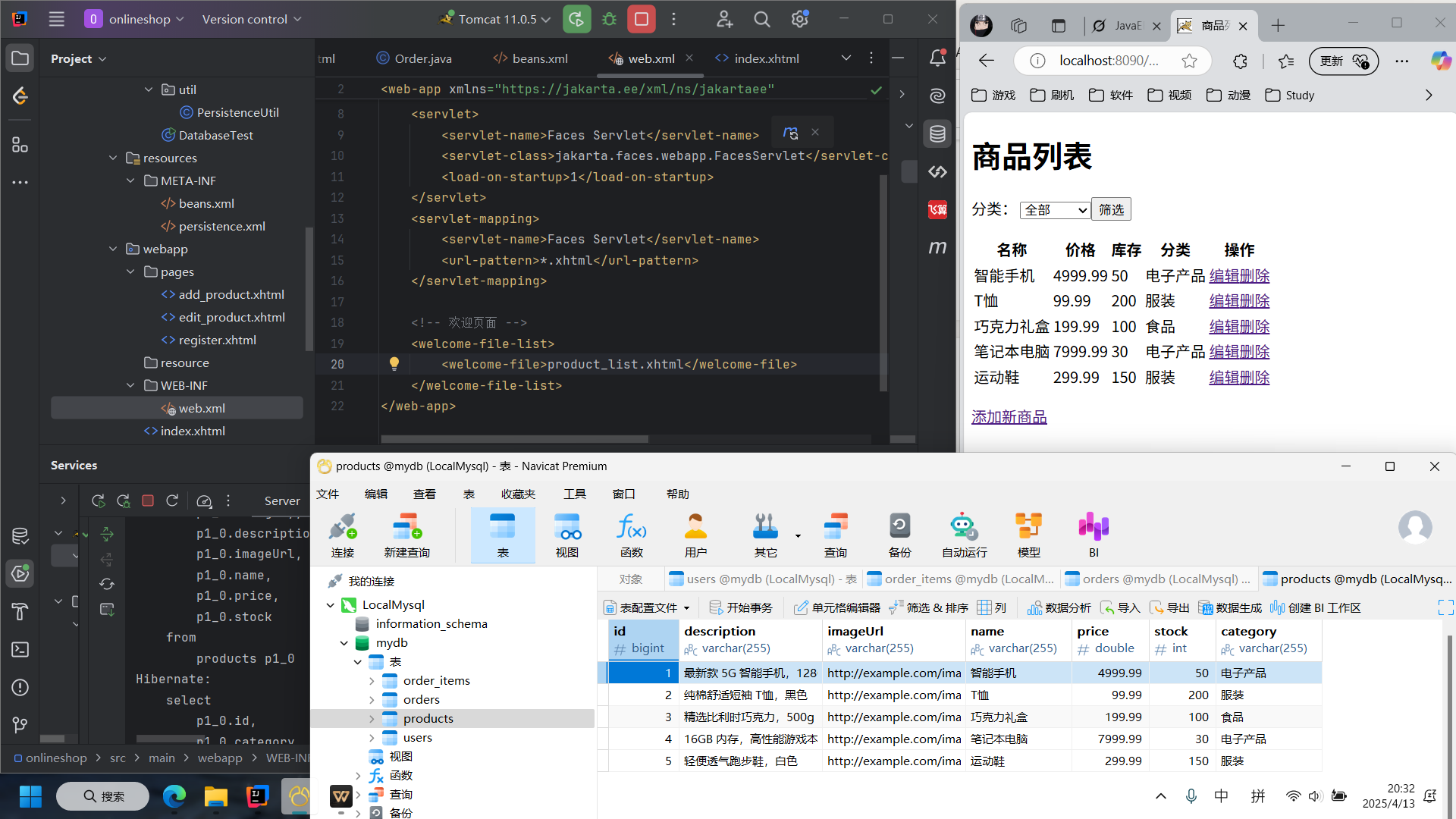


支持商品的添加



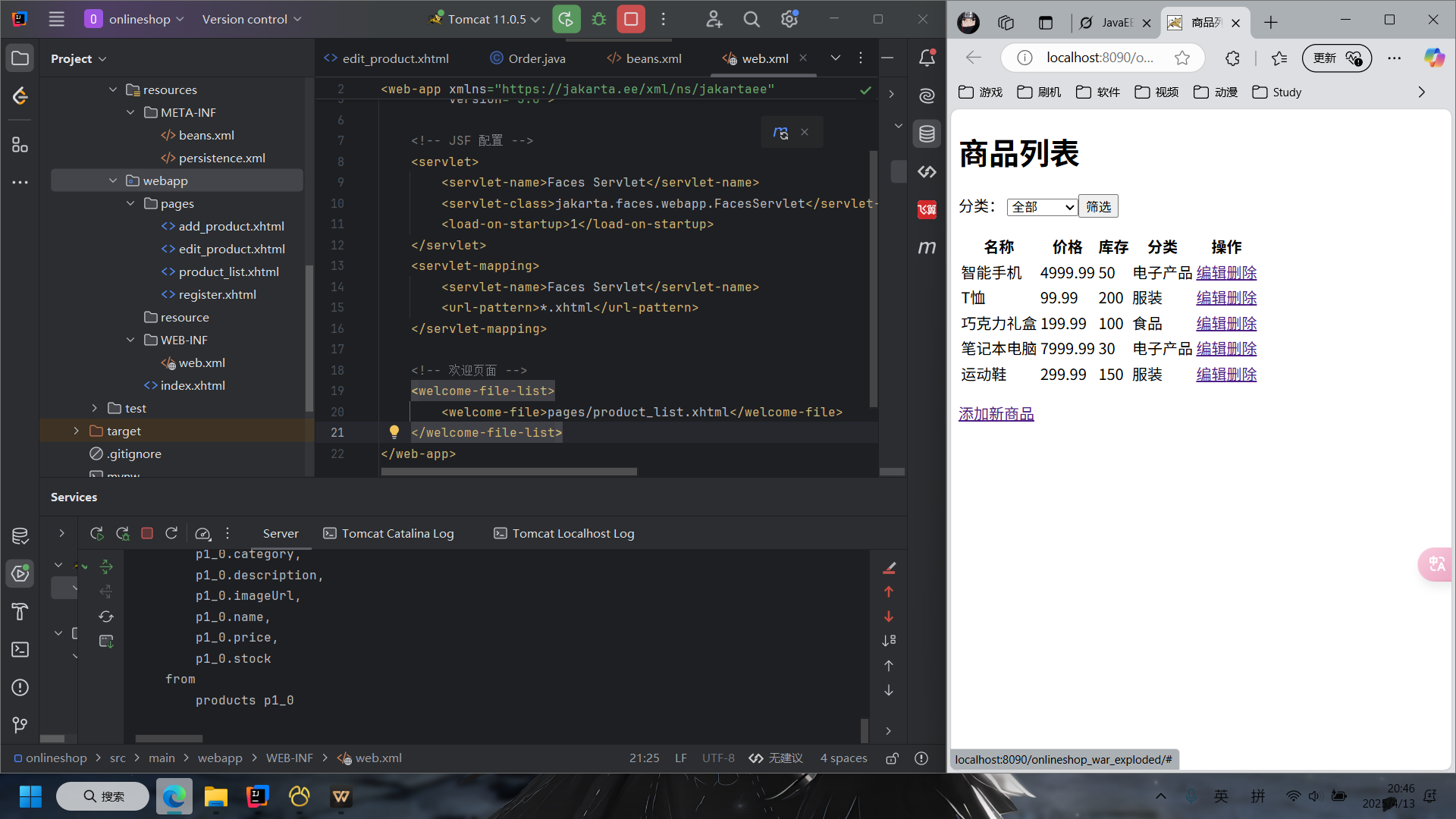
支持商品删除（删除 最后一条商品信息）

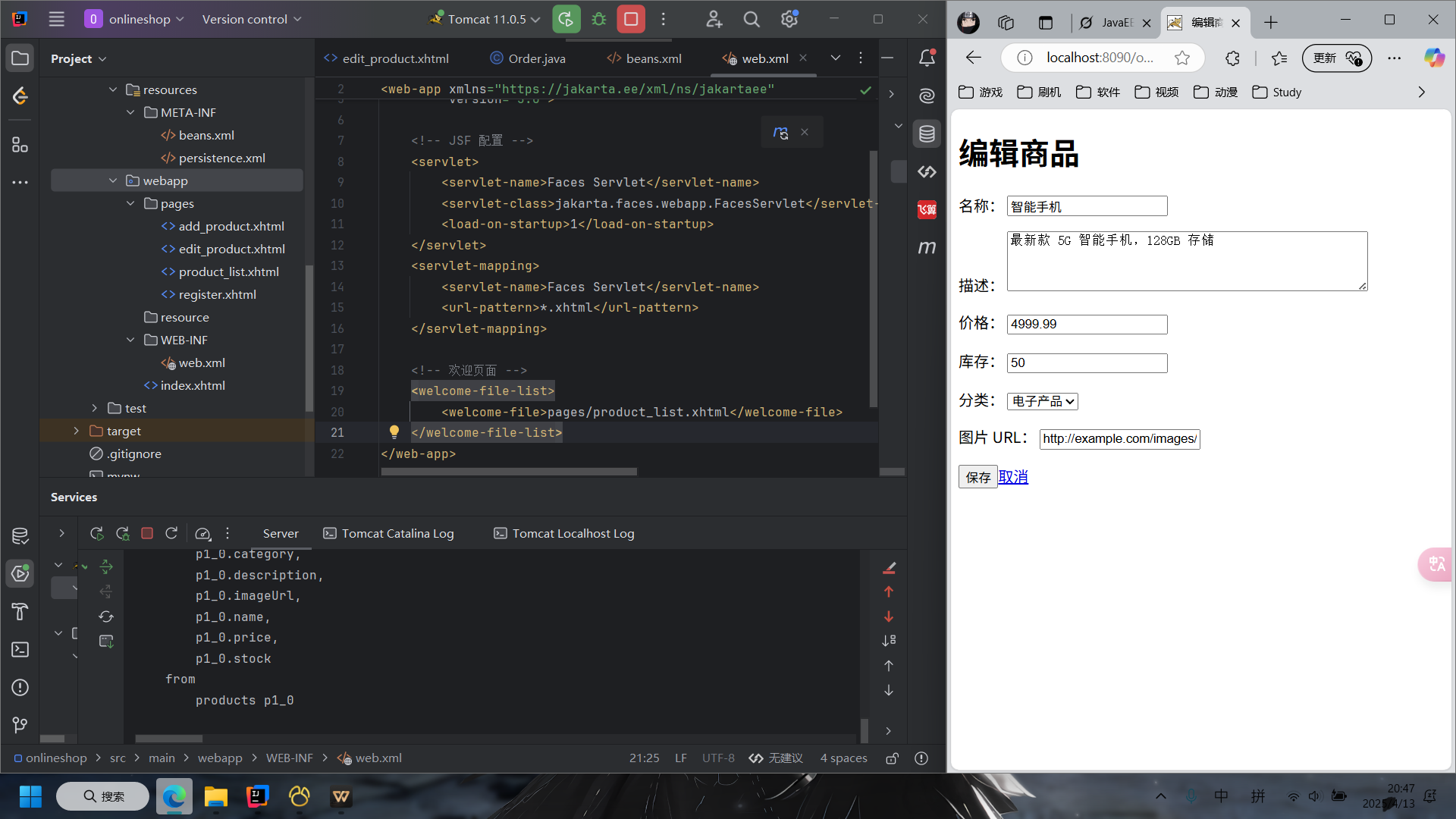
**Before：**

**Later：**

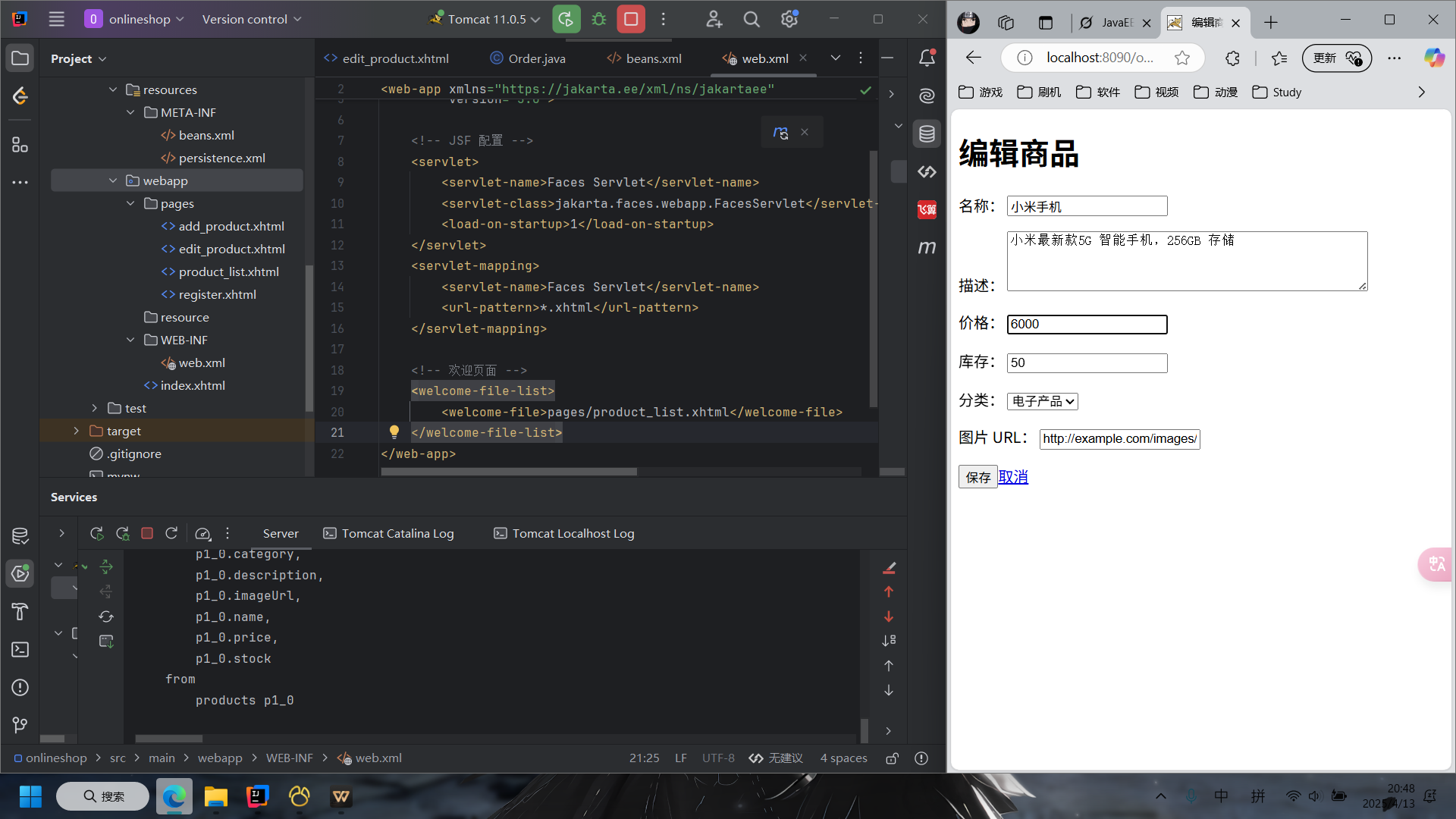
支持商品编辑（实现对商品信息的编辑修改功能）

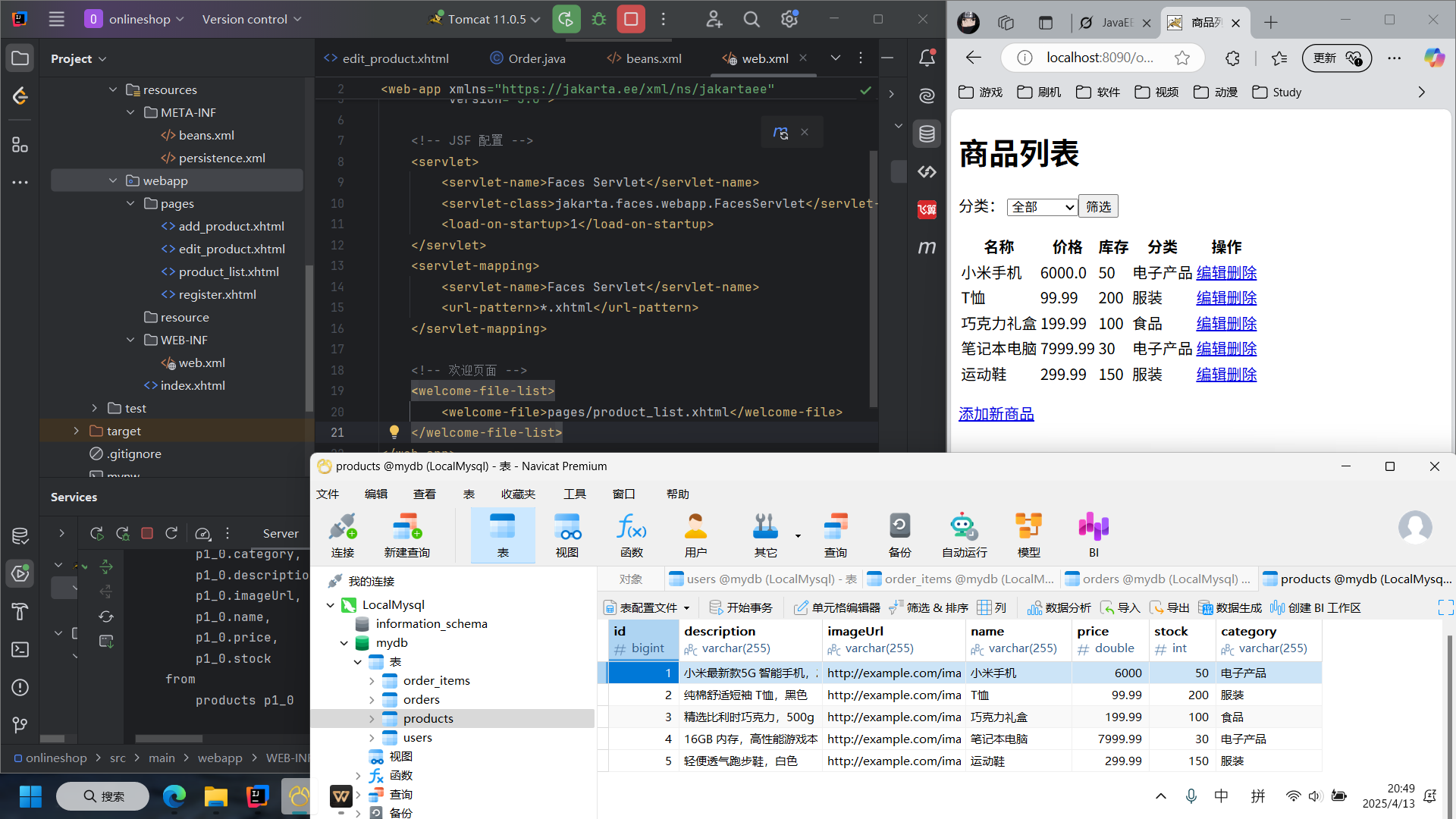
**Before：**





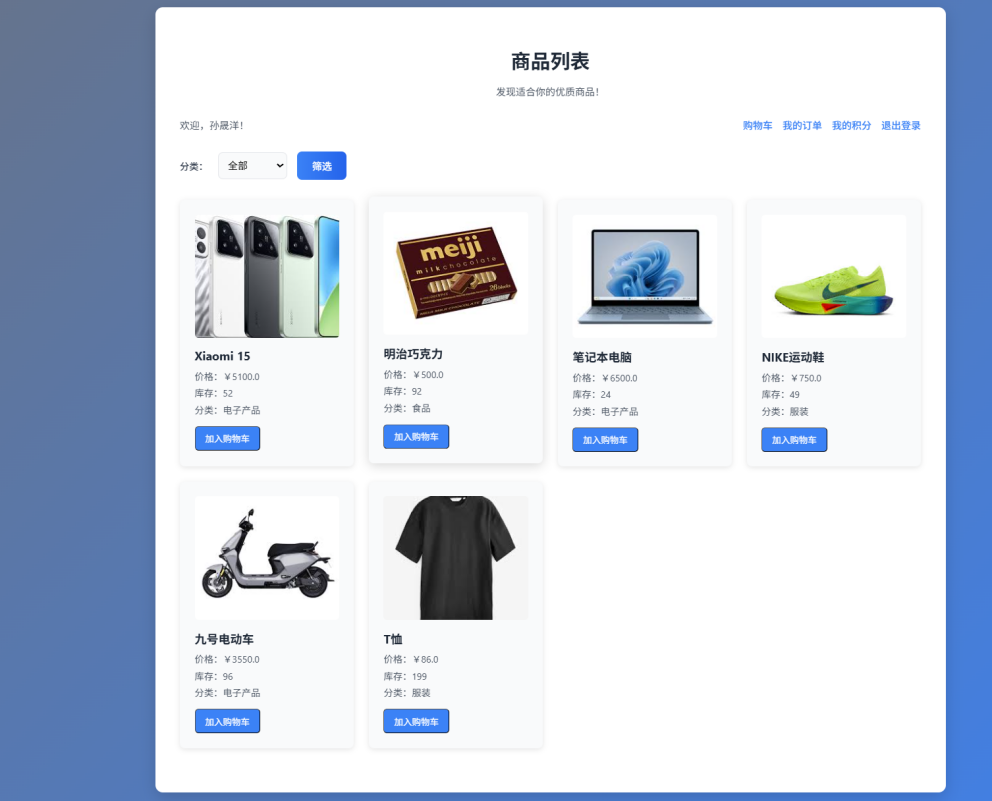
**Later：**





实现订单管理模块（购物车、订单创建、订单查询）

1. 将商品“小米”添加购物车



②用户可使用积分抵扣并显示 折扣优惠，实付价格



③点击提交订单 跳转订单界面 选择“支付/取消”

④积分界面实现积分转换（10元：1积分）



整合、修复项目

整体系统集成与优化，测试与修复 Bug，撰写项目文档，最终确认系统功能完整性，并提交项目

**五、分工**

角色 主要职责

孙彬祺 后端开发（JPA、CDI），数据库设计，API 开发

林兴博 前端开发（JSF），页面设计，用户体验优化，测试与调试

## 六、项目成果

* **技术栈**：JavaEE、JSF、JPA (Hibernate)、CDI、MySQL、Tomcat。
* **文件结构**：
  + **根目录**：pom.xml（Maven 配置）。
  + **实体**：User, Product, PointRecord, CartItem, Order, OrderItem。
  + **服务**：UserService, OrderService, ProductService。
  + **工具**：PersistenceUtil, LocalDateTimeConverter。
  + **控制器**：AuthController, OrderController, ProductController, UserController, PointController。
  + **页面**：login.xhtml, product\_list.xhtml, user\_management.xhtml, register.xhtml, cart.xhtml, order\_list.xhtml, points.xhtml, edit\_product.xhtml, add\_product.xhtml。
  + **配置**：web.xml, faces-config.xml, sun-jaxws.xml, persistence.xml。
  + **资源**：resources/images/（图片）。
  + **已删除**：product\_list\_user.xhtml（功能整合至 product\_list.xhtml）。
* **数据库**：
  + 表：users, products, point\_records, orders, order\_items。
  + 示例数据：管理员（admin）、用户（user1, user2）、商品（手机, T恤）、订单、购物车项。
* **权限控制**：管理员功能（rendered="#{authController.isLoggedIn() and authController.currentUser.role == 'ADMIN'}"）。

## 七、部署说明

* **环境**：Tomcat 9/10，MySQL，JNDI 数据源 jdbc/onlineshop。
* **步骤**：
  1. 配置 persistence.xml 数据源。
  2. 编译部署
  3. 访问：http://localhost:8090/onlineshop\_war\_exploded/

## 总结

**项目遇到的问题：**

· **注册页面属性绑定错误**：

* **问题**：register.xhtml 报 jakarta.el.PropertyNotFoundException，提示 UserController 缺少 user 属性，导致无法绑定表单字段（如 userController.user.username）。
* **解决**：在 UserController.java 中添加 private User user = new User() 和 confirmPassword 属性，完善 getter/setter，并实现 register 方法调用 UserService.registerUser，修复绑定问题。

· **实体字段类型不匹配**：

* **问题**：UserController 的 register 方法中，user.setPoints(0L) 和 user.setRole("USER") 报类型错误，因 User 实体定义 points 为 int、 role 为 User.Role 枚举，而非 long 和 String。
* **解决**：修改 UserController 使用 user.setPoints(0) 和 user.setRole(User.Role.USER)，并提供 User.java 定义 Role 枚举（ADMIN, USER），确保类型一致。

· **冗余文件导致功能分散**：

* **问题**：初始有两个商品列表页面（product\_list.xhtml 和 product\_list\_user.xhtml），功能重复，维护复杂。
* **解决**：删除 product\_list\_user.xhtml（artifact\_id: 3f2df01e-6524-4081-991f-0bc4993631af），将用户浏览和购物车功能合并到 product\_list.xhtml，通过 rendered 属性实现管理员和用户权限控制。

项目成功实现基于 JavaEE 的在线商城系统，涵盖用户管理、商品管理、订单管理和积分系统。系统通过 JSF 提供友好的 Web 界面，JPA 确保数据高效管理，CDI 增强模块化。所有功能按计划完成，测试通过，文档齐全。建议备份代码和数据库，记录部署步骤以便维护。

**九、附录**

代码

1. 控制器类

**AuthController.java**

package com.example.onlineshop.controller;

import com.example.onlineshop.entity.User;

import com.example.onlineshop.service.UserService;

import jakarta.enterprise.context.SessionScoped;

import jakarta.faces.application.FacesMessage;

import jakarta.faces.context.FacesContext;

import jakarta.inject.Inject;

import jakarta.inject.Named;

import java.io.Serializable;

@Named

@SessionScoped

public class AuthController implements Serializable {

private User currentUser;

private String username;

private String password;

@Inject

private UserService userService;

public String login() {

try {

currentUser = userService.authenticate(username, password);

if (currentUser != null) {

if ("USER".equals(currentUser.getRole())) {

FacesContext.getCurrentInstance().addMessage(null,

new FacesMessage(FacesMessage.SEVERITY\_ERROR, "权限不足", "仅管理员可访问"));

currentUser = null;

return "login?faces-redirect=true";

}

return "product\_list?faces-redirect=true";

} else {

FacesContext.getCurrentInstance().addMessage(null,

new FacesMessage(FacesMessage.SEVERITY\_ERROR, "登录失败", "用户名或密码错误"));

return "login?faces-redirect=true";

}

} catch (Exception e) {

FacesContext.getCurrentInstance().addMessage(null,

new FacesMessage(FacesMessage.SEVERITY\_ERROR, "登录错误", "系统错误，请稍后重试"));

return "login?faces-redirect=true";

}

}

public String logout() {

currentUser = null;

username = null;

password = null;

return "login?faces-redirect=true";

}

public boolean isLoggedIn() {

return currentUser != null;

}

public User getCurrentUser() {

return currentUser;

}

public void setCurrentUser(User currentUser) {

this.currentUser = currentUser;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

}

**OrderController.java**

package com.example.onlineshop.controller;

import com.example.onlineshop.entity.CartItem;

import com.example.onlineshop.entity.Order;

import com.example.onlineshop.entity.Product;

import com.example.onlineshop.entity.User;

import com.example.onlineshop.service.OrderService;

import com.example.onlineshop.service.ProductService;

import com.example.onlineshop.service.UserService;

import jakarta.enterprise.context.SessionScoped;

import jakarta.faces.application.FacesMessage;

import jakarta.faces.context.FacesContext;

import jakarta.inject.Inject;

import jakarta.inject.Named;

import java.io.Serializable;

import java.util.ArrayList;

import java.util.List;

@Named

@SessionScoped

public class OrderController implements Serializable {

@Inject private UserService userService;

@Inject private ProductService productService;

@Inject private OrderService orderService;

@Inject private AuthController authController;

private String username;

private String password;

private List<CartItem> cartItems = new ArrayList<>();

private int pointsToUse;

private double pointsDiscount;

public String login() {

User user = userService.findByUsername(username);

if (user != null && user.getPassword().equals(password)) {

if ("USER".equals(user.getRole())) {

FacesContext.getCurrentInstance().addMessage(null,

new FacesMessage(FacesMessage.SEVERITY\_ERROR, "权限不足", "仅管理员可访问"));

return "login?faces-redirect=true";

}

authController.setCurrentUser(user);

System.out.println("login: username=" + username + ", points=" + user.getPoints());

return "product\_list?faces-redirect=true";

}

FacesContext.getCurrentInstance().addMessage(null,

new FacesMessage(FacesMessage.SEVERITY\_ERROR, "用户名或密码错误", null));

return null;

}

public void addToCart(Long productId, int quantity) {

Product product = productService.findProductById(productId);

if (product != null && product.getStock() >= quantity) {

for (CartItem item : cartItems) {

if (item.getProduct().getId().equals(productId)) {

item.setQuantity(item.getQuantity() + quantity);

FacesContext.getCurrentInstance().addMessage(null,

new FacesMessage(FacesMessage.SEVERITY\_INFO, "已更新购物车：" + product.getName(), null));

return;

}

}

cartItems.add(new CartItem(product, quantity));

FacesContext.getCurrentInstance().addMessage(null,

new FacesMessage(FacesMessage.SEVERITY\_INFO, "已加入购物车：" + product.getName(), null));

} else {

FacesContext.getCurrentInstance().addMessage(null,

new FacesMessage(FacesMessage.SEVERITY\_ERROR, "库存不足或商品不存在", null));

}

}

public String updateCartItem(Long productId, int quantity) {

for (CartItem item : cartItems) {

if (item.getProduct().getId().equals(productId)) {

if (quantity <= 0) {

cartItems.remove(item);

} else {

Product product = productService.findProductById(productId);

if (product != null && product.getStock() >= quantity) {

item.setQuantity(quantity);

} else {

FacesContext.getCurrentInstance().addMessage(null,

new FacesMessage(FacesMessage.SEVERITY\_ERROR, "库存不足", null));

return null;

}

}

break;

}

}

return "cart?faces-redirect=true";

}

public String removeFromCart(Long productId) {

cartItems.removeIf(item -> item.getProduct().getId().equals(productId));

return "cart?faces-redirect=true";

}

public String applyPoints() {

pointsToUse = Math.max(0, pointsToUse);

if (pointsToUse == 0) {

pointsDiscount = 0;

FacesContext.getCurrentInstance().addMessage(null,

new FacesMessage(FacesMessage.SEVERITY\_INFO, "已移除积分抵扣", null));

return null;

}

if (!authController.isLoggedIn()) {

FacesContext.getCurrentInstance().addMessage(null,

new FacesMessage(FacesMessage.SEVERITY\_ERROR, "请先登录", null));

return null;

}

User user = authController.getCurrentUser();

if (pointsToUse > user.getPoints()) {

pointsToUse = 0;

pointsDiscount = 0;

FacesContext.getCurrentInstance().addMessage(null,

new FacesMessage(FacesMessage.SEVERITY\_ERROR, "积分不足", null));

return null;

}

pointsDiscount = pointsToUse / 100.0;

FacesContext.getCurrentInstance().addMessage(null,

new FacesMessage(FacesMessage.SEVERITY\_INFO, "积分抵扣应用成功", null));

return null;

}

public String submitOrder() {

if (!authController.isLoggedIn()) {

FacesContext.getCurrentInstance().addMessage(null,

new FacesMessage(FacesMessage.SEVERITY\_ERROR, "请先登录", null));

return "login?faces-redirect=true";

}

if (cartItems.isEmpty()) {

FacesContext.getCurrentInstance().addMessage(null,

new FacesMessage(FacesMessage.SEVERITY\_ERROR, "购物车为空", null));

return null;

}

try {

pointsToUse = Math.max(0, pointsToUse);

orderService.createOrder(authController.getCurrentUser(), cartItems, pointsToUse);

cartItems.clear();

pointsToUse = 0;

pointsDiscount = 0;

FacesContext.getCurrentInstance().addMessage(null,

new FacesMessage(FacesMessage.SEVERITY\_INFO, "订单提交成功！", null));

return "order\_list?faces-redirect=true";

} catch (Exception e) {

FacesContext.getCurrentInstance().addMessage(null,

new FacesMessage(FacesMessage.SEVERITY\_ERROR, "订单提交失败：" + e.getMessage(), null));

return null;

}

}

public String payOrder(Long orderId) {

if (!authController.isLoggedIn()) {

FacesContext.getCurrentInstance().addMessage(null,

new FacesMessage(FacesMessage.SEVERITY\_ERROR, "请先登录", null));

return "login?faces-redirect=true";

}

try {

orderService.payOrder(orderId, authController.getCurrentUser());

FacesContext.getCurrentInstance().addMessage(null,

new FacesMessage(FacesMessage.SEVERITY\_INFO, "订单支付成功", null));

return "order\_list?faces-redirect=true";

} catch (Exception e) {

FacesContext.getCurrentInstance().addMessage(null,

new FacesMessage(FacesMessage.SEVERITY\_ERROR, "支付失败：" + e.getMessage(), null));

return "order\_list?faces-redirect=true";

}

}

public String cancelOrder(Long orderId) {

if (!authController.isLoggedIn()) {

FacesContext.getCurrentInstance().addMessage(null,

new FacesMessage(FacesMessage.SEVERITY\_ERROR, "请先登录", null));

return "login?faces-redirect=true";

}

try {

orderService.cancelOrder(orderId);

FacesContext.getCurrentInstance().addMessage(null,

new FacesMessage(FacesMessage.SEVERITY\_INFO, "订单已取消", null));

return "order\_list?faces-redirect=true";

} catch (Exception e) {

FacesContext.getCurrentInstance().addMessage(null,

new FacesMessage(FacesMessage.SEVERITY\_ERROR, "取消失败：" + e.getMessage(), null));

return "order\_list?faces-redirect=true";

}

}

public String goToCart() {

if (!authController.isLoggedIn()) {

FacesContext.getCurrentInstance().addMessage(null,

new FacesMessage(FacesMessage.SEVERITY\_ERROR, "请先登录", null));

return "login?faces-redirect=true";

}

return "cart?faces-redirect=true";

}

public String goToOrders() {

if (!authController.isLoggedIn()) {

FacesContext.getCurrentInstance().addMessage(null,

new FacesMessage(FacesMessage.SEVERITY\_ERROR, "请先登录", null));

return "login?faces-redirect=true";

}

return "order\_list?faces-redirect=true";

}

public String goToProductList() {

return "product\_list?faces-redirect=true";

}

public double getCartTotal() {

double total = cartItems.stream()

.mapToDouble(item -> item.getQuantity() \* item.getProduct().getPrice())

.sum();

return Math.max(0, total - pointsDiscount);

}

public String getUsername() { return username; }

public void setUsername(String username) { this.username = username; }

public String getPassword() { return password; }

public void setPassword(String password) { this.password = password; }

public User getCurrentUser() { return authController.getCurrentUser(); }

public void setCurrentUser(User currentUser) { authController.setCurrentUser(currentUser); }

public List<CartItem> getCartItems() { return cartItems; }

public void setCartItems(List<CartItem> cartItems) { this.cartItems = cartItems; }

public int getPointsToUse() { return pointsToUse; }

public void setPointsToUse(int pointsToUse) { this.pointsToUse = pointsToUse; }

public double getPointsDiscount() { return pointsDiscount; }

public void setPointsDiscount(double pointsDiscount) { this.pointsDiscount = pointsDiscount; }

public List<Order> getOrders() {

if (!authController.isLoggedIn()) {

return new ArrayList<>();

}

return orderService.findOrdersByUser(authController.getCurrentUser().getId());

}

}

**PointController.java**

package com.example.onlineshop.controller;

import com.example.onlineshop.entity.PointRecord;

import com.example.onlineshop.service.PointService;

import jakarta.enterprise.context.SessionScoped;

import jakarta.inject.Inject;

import jakarta.inject.Named;

import java.io.Serializable;

import java.util.Collections;

import java.util.List;

@Named

@SessionScoped

public class PointController implements Serializable {

@Inject

private PointService pointService;

@Inject

private AuthController authController;

public List<PointRecord> getPointRecords() {

if (!authController.isLoggedIn()) {

return Collections.emptyList();

}

return pointService.findPointRecordsByUser(authController.getCurrentUser());

}

public String goToPoints() {

if (!authController.isLoggedIn()) {

return "login?faces-redirect=true";

}

return "points?faces-redirect=true";

}

}

**ProductController.java**

package com.example.onlineshop.controller;

import com.example.onlineshop.entity.Product;

import com.example.onlineshop.service.ProductService;

import jakarta.enterprise.context.SessionScoped;

import jakarta.inject.Inject;

import jakarta.inject.Named;

import java.io.Serializable;

import java.util.List;

@Named

@SessionScoped

public class ProductController implements Serializable {

@Inject

private ProductService productService;

private Product product = new Product(); // 用于添加或编辑商品

private Long editProductId; // 用于编辑时保存商品 ID

private String selectedCategory; // 用于分类筛选

// 添加商品

public String addProduct() {

productService.addProduct(product);

product = new Product(); // 清空表单

return "product\_list?faces-redirect=true";

}

// 开始编辑商品

public String editProduct(Long id) {

this.editProductId = id;

this.product = productService.findProductById(id);

if (this.product == null) {

// 如果商品不存在，返回列表页面

return "product\_list?faces-redirect=true";

}

return "edit\_product?faces-redirect=true";

}

// 保存编辑

public String saveEdit() {

productService.updateProduct(product);

product = new Product(); // 清空表单

return "product\_list?faces-redirect=true";

}

// 删除商品

public String deleteProduct(Long id) {

productService.deleteProduct(id);

return "product\_list?faces-redirect=true";

}

// 获取所有商品

public List<Product> getAllProducts() {

return productService.findAllProducts();

}

// 按分类获取商品

public List<Product> getProductsByCategory() {

if (selectedCategory == null || selectedCategory.isEmpty()) {

return getAllProducts();

}

return productService.findProductsByCategory(selectedCategory);

}

public String goToAddProduct() {

// 创建一个新的Product对象

product = new Product();

// 返回add\_product页面，并重定向

return "add\_product?faces-redirect=true";

}

public String cancelAddProduct() {

product = new Product(); // 清空表单

return "product\_list?faces-redirect=true";

}

// Getters and Setters

public Product getProduct() { return product; }

public void setProduct(Product product) { this.product = product; }

public Long getEditProductId() { return editProductId; }

public void setEditProductId(Long editProductId) { this.editProductId = editProductId; }

public String getSelectedCategory() { return selectedCategory; }

public void setSelectedCategory(String selectedCategory) { this.selectedCategory = selectedCategory; }

}

**UserController.java**

package com.example.onlineshop.controller;

import com.example.onlineshop.entity.User;

import com.example.onlineshop.service.UserService;

import jakarta.enterprise.context.SessionScoped;

import jakarta.faces.application.FacesMessage;

import jakarta.faces.context.FacesContext;

import jakarta.inject.Inject;

import jakarta.inject.Named;

import java.io.Serializable;

import java.util.List;

@Named

@SessionScoped

public class UserController implements Serializable {

@Inject

private UserService userService;

private List<User> users;

private User selectedUser;

public void init() {

users = userService.findAllUsers();

selectedUser = null;

}

public String editUser(Long userId) {

selectedUser = userService.findById(userId);

if (selectedUser == null) {

FacesContext.getCurrentInstance().addMessage(null,

new FacesMessage(FacesMessage.SEVERITY\_ERROR, "错误", "用户不存在"));

return null;

}

return null; // 留在当前页面

}

public String saveUser() {

try {

userService.updateUser(selectedUser);

FacesContext.getCurrentInstance().addMessage(null,

new FacesMessage(FacesMessage.SEVERITY\_INFO, "成功", "用户信息已更新"));

init(); // 刷新用户列表

return null;

} catch (Exception e) {

FacesContext.getCurrentInstance().addMessage(null,

new FacesMessage(FacesMessage.SEVERITY\_ERROR, "错误", "更新用户失败: " + e.getMessage()));

return null;

}

}

public String cancelEdit() {

selectedUser = null;

return null;

}

public List<User> getUsers() {

if (users == null) {

init();

}

return users;

}

public void setUsers(List<User> users) {

this.users = users;

}

public User getSelectedUser() {

return selectedUser;

}

public void setSelectedUser(User selectedUser) {

this.selectedUser = selectedUser;

}

}

1. 实体类

**Cartltem.java**

package com.example.onlineshop.entity;

import java.io.Serializable;

public class CartItem implements Serializable {

private Product product;

private int quantity;

public CartItem() {

}

public CartItem(Product product, int quantity) {

this.product = product;

this.quantity = quantity;

}

// Getters and Setters

public Product getProduct() { return product; }

public void setProduct(Product product) { this.product = product; }

public int getQuantity() { return quantity; }

public void setQuantity(int quantity) { this.quantity = quantity; }

}

**Order.java**

package com.example.onlineshop.entity;

import jakarta.persistence.\*;

import java.io.Serializable;

import java.time.LocalDateTime;

import java.util.ArrayList;

import java.util.List;

@Entity

@Table(name = "orders")

public class Order implements Serializable {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@ManyToOne

@JoinColumn(name = "user\_id", nullable = false)

private User user;

@Column(nullable = false)

private LocalDateTime orderDate;

@Column(nullable = false)

private double totalAmount;

@Column(nullable = false)

private String status; // PENDING, PAID, CANCELLED

private Integer pointsUsed;

@OneToMany(mappedBy = "order", cascade = CascadeType.ALL, orphanRemoval = true)

private List<OrderItem> orderItems = new ArrayList<>();

public Long getId() { return id; }

public void setId(Long id) { this.id = id; }

public User getUser() { return user; }

public void setUser(User user) { this.user = user; }

public LocalDateTime getOrderDate() { return orderDate; }

public void setOrderDate(LocalDateTime orderDate) { this.orderDate = orderDate; }

public double getTotalAmount() { return totalAmount; }

public void setTotalAmount(double totalAmount) { this.totalAmount = totalAmount; }

public String getStatus() { return status; }

public void setStatus(String status) { this.status = status; }

public Integer getPointsUsed() { return pointsUsed; }

public void setPointsUsed(Integer pointsUsed) { this.pointsUsed = pointsUsed; }

public List<OrderItem> getOrderItems() { return orderItems; }

public void setOrderItems(List<OrderItem> orderItems) { this.orderItems = orderItems; }

}

**Orderltem.java**

package com.example.onlineshop.entity;

import jakarta.persistence.\*;

@Entity

@Table(name = "order\_items")

public class OrderItem {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@ManyToOne

@JoinColumn(name = "product\_id", nullable = false)

private Product product;

@Column(nullable = false)

private Integer quantity;

@Column(nullable = false)

private Double price;

@ManyToOne

@JoinColumn(name = "order\_id", nullable = false)

private Order order;

// Getters and Setters

public Long getId() { return id; }

public void setId(Long id) { this.id = id; }

public Product getProduct() { return product; }

public void setProduct(Product product) { this.product = product; }

public Integer getQuantity() { return quantity; }

public void setQuantity(Integer quantity) { this.quantity = quantity; }

public Double getPrice() { return price; }

public void setPrice(Double price) { this.price = price; }

public Order getOrder() { return order; }

public void setOrder(Order order) { this.order = order; }

}

**PointRecord.java**

package com.example.onlineshop.entity;

import jakarta.persistence.\*;

import java.time.LocalDateTime;

@Entity

@Table(name = "point\_records")

public class PointRecord {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@ManyToOne

@JoinColumn(name = "user\_id", nullable = false)

private User user;

@Column(nullable = false)

private int points;

@Column(nullable = false)

private String source;

@Column(name = "created\_at", nullable = false)

private LocalDateTime createdAt;

// Getters and Setters

public Long getId() { return id; }

public void setId(Long id) { this.id = id; }

public User getUser() { return user; }

public void setUser(User user) { this.user = user; }

public int getPoints() { return points; }

public void setPoints(int points) { this.points = points; }

public String getSource() { return source; }

public void setSource(String source) { this.source = source; }

public LocalDateTime getCreatedAt() { return createdAt; }

public void setCreatedAt(LocalDateTime createdAt) { this.createdAt = createdAt; }

}

**Product.java**

package com.example.onlineshop.entity;

import jakarta.persistence.\*;

@Entity

@Table(name = "products")

public class Product {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(nullable = false)

private String name;

private String description;

@Column(nullable = false)

private Double price;

private String imageUrl;

@Column(nullable = false)

private Integer stock;

@Column(nullable = false)

private String category; // 新增分类字段，例如 "电子产品"、"服装"等

// Getters and Setters

public Long getId() { return id; }

public void setId(Long id) { this.id = id; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

public String getDescription() { return description; }

public void setDescription(String description) { this.description = description; }

public Double getPrice() { return price; }

public void setPrice(Double price) { this.price = price; }

public String getImageUrl() { return imageUrl; }

public void setImageUrl(String imageUrl) { this.imageUrl = imageUrl; }

public Integer getStock() { return stock; }

public void setStock(Integer stock) { this.stock = stock; }

public String getCategory() { return category; }

public void setCategory(String category) { this.category = category; }

}

**User.java**

package com.example.onlineshop.entity;

import jakarta.persistence.\*;

@Entity

@Table(name = "users")

public class User {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(nullable = false, unique = true)

private String username;

@Column(nullable = false)

private String password;

@Column(nullable = false)

private int points;

@Enumerated(EnumType.STRING)

@Column(nullable = false)

private Role role;

@Column

private String nickname;

@Column

private String avatarUrl;

public String getAvatarUrl() { return avatarUrl; }

public void setAvatarUrl(String avatarUrl) { this.avatarUrl = avatarUrl; }

public enum Role { USER, ADMIN }

// Getters and Setters

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public int getPoints() {

return points;

}

public void setPoints(int points) {

this.points = points;

}

public Role getRole() {

return role;

}

public void setRole(Role role) {

this.role = role;

}

public String getNickname() {

return nickname;

}

public void setNickname(String nickname) {

this.nickname = nickname;

}

}

1. 服务类

**OrderService.java**

package com.example.onlineshop.service;

import com.example.onlineshop.entity.\*;

import com.example.onlineshop.util.PersistenceUtil;

import jakarta.enterprise.context.ApplicationScoped;

import jakarta.inject.Inject;

import jakarta.persistence.EntityManager;

import jakarta.transaction.Transactional;

import java.io.Serializable;

import java.time.LocalDateTime;

import java.util.List;

import java.util.logging.Logger;

@ApplicationScoped

public class OrderService implements Serializable {

private static final Logger LOGGER = Logger.getLogger(OrderService.class.getName());

@Inject

private PersistenceUtil persistenceUtil;

@Transactional

public void createOrder(User user, List<CartItem> cart, int pointsUsed) {

EntityManager em = persistenceUtil.getEntityManager();

try {

em.getTransaction().begin();

Order order = new Order();

order.setUser(user);

order.setOrderDate(LocalDateTime.now());

order.setStatus("PENDING");

double totalAmount = cart.stream()

.mapToDouble(item -> item.getProduct().getPrice() \* item.getQuantity())

.sum();

// 积分抵扣：100积分 = 1元

pointsUsed = Math.max(0, pointsUsed); // 确保非负

if (pointsUsed > 0) {

if (pointsUsed > user.getPoints()) {

throw new RuntimeException("积分不足");

}

double discount = pointsUsed / 100.0;

totalAmount -= discount;

order.setPointsUsed(pointsUsed);

} else {

order.setPointsUsed(0); // 明确设置为 0

}

order.setTotalAmount(Math.max(0, totalAmount));

for (CartItem item : cart) {

Product product = em.find(Product.class, item.getProduct().getId());

if (product == null) {

throw new RuntimeException("商品 ID " + item.getProduct().getId() + " 不存在");

}

if (product.getStock() < item.getQuantity()) {

throw new RuntimeException("商品 " + product.getName() + " 库存不足");

}

product.setStock(product.getStock() - item.getQuantity());

OrderItem orderItem = new OrderItem();

orderItem.setProduct(product);

orderItem.setQuantity(item.getQuantity());

orderItem.setPrice(product.getPrice());

orderItem.setOrder(order);

order.getOrderItems().add(orderItem);

em.merge(product);

}

em.persist(order);

em.getTransaction().commit();

LOGGER.info("订单创建成功: orderId=" + order.getId() + ", userId=" + user.getId());

} catch (Exception e) {

if (em.getTransaction().isActive()) {

em.getTransaction().rollback();

}

LOGGER.severe("创建订单失败: " + e.getMessage());

throw new RuntimeException("创建订单失败: " + e.getMessage());

} finally {

em.close();

}

}

public List<Order> findOrdersByUser(Long userId) {

EntityManager em = persistenceUtil.getEntityManager();

try {

return em.createQuery(

"SELECT o FROM Order o JOIN FETCH o.orderItems WHERE o.user.id = :userId ORDER BY o.orderDate DESC",

Order.class)

.setParameter("userId", userId)

.getResultList();

} finally {

em.close();

}

}

@Transactional

public void payOrder(Long orderId, User user) {

EntityManager em = persistenceUtil.getEntityManager();

try {

em.getTransaction().begin();

Order order = em.find(Order.class, orderId);

if (order == null) {

throw new RuntimeException("订单不存在");

}

if (!order.getUser().getId().equals(user.getId())) {

throw new RuntimeException("无权限支付此订单");

}

if (!order.getStatus().equals("PENDING")) {

throw new RuntimeException("订单状态不支持支付");

}

// 扣除积分（如果有）

int pointsUsed = order.getPointsUsed() != null ? order.getPointsUsed() : 0;

if (pointsUsed > 0) {

if (pointsUsed > user.getPoints()) {

throw new RuntimeException("积分不足");

}

user.setPoints(user.getPoints() - pointsUsed);

PointRecord pointRecord = new PointRecord();

pointRecord.setUser(user);

pointRecord.setPoints(-pointsUsed);

pointRecord.setSource("订单积分抵扣");

pointRecord.setCreatedAt(LocalDateTime.now());

em.persist(pointRecord);

}

// 增加积分：10元 = 1积分

int pointsEarned = (int) (order.getTotalAmount() / 10);

if (pointsEarned > 0) {

user.setPoints(user.getPoints() + pointsEarned);

PointRecord pointRecord = new PointRecord();

pointRecord.setUser(user);

pointRecord.setPoints(pointsEarned);

pointRecord.setSource("订单支付");

pointRecord.setCreatedAt(LocalDateTime.now());

em.persist(pointRecord);

}

// 更新订单状态

order.setStatus("PAID");

em.merge(user);

em.merge(order);

em.getTransaction().commit();

LOGGER.info("订单支付成功: orderId=" + orderId + ", userId=" + user.getId() + ", pointsUsed=" + pointsUsed);

} catch (Exception e) {

if (em.getTransaction().isActive()) {

em.getTransaction().rollback();

}

LOGGER.severe("订单支付失败: orderId=" + orderId + ", error=" + e.getMessage());

throw new RuntimeException("订单支付失败: " + e.getMessage());

} finally {

em.close();

}

}

@Transactional

public void cancelOrder(Long orderId) {

EntityManager em = persistenceUtil.getEntityManager();

try {

em.getTransaction().begin();

Order order = em.find(Order.class, orderId);

if (order == null) {

throw new RuntimeException("订单不存在");

}

if (!order.getStatus().equals("PENDING")) {

throw new RuntimeException("订单状态不支持取消");

}

order.setStatus("CANCELLED");

em.merge(order);

em.getTransaction().commit();

LOGGER.info("订单取消成功: orderId=" + orderId);

} catch (Exception e) {

if (em.getTransaction().isActive()) {

em.getTransaction().rollback();

}

LOGGER.severe("订单取消失败: " + e.getMessage());

throw new RuntimeException("订单取消失败: " + e.getMessage());

} finally {

em.close();

}

}

@Transactional

public void updateOrderStatus(Long orderId, String status) {

EntityManager em = persistenceUtil.getEntityManager();

try {

em.getTransaction().begin();

Order order = em.find(Order.class, orderId);

if (order == null) {

throw new RuntimeException("订单不存在");

}

order.setStatus(status);

em.merge(order);

em.getTransaction().commit();

LOGGER.info("订单状态更新成功: orderId=" + orderId + ", status=" + status);

} catch (Exception e) {

if (em.getTransaction().isActive()) {

em.getTransaction().rollback();

}

LOGGER.severe("更新订单状态失败: " + e.getMessage());

throw new RuntimeException("更新订单状态失败: " + e.getMessage());

} finally {

em.close();

}

}

}

**PointService.java**

package com.example.onlineshop.service;

import com.example.onlineshop.entity.PointRecord;

import com.example.onlineshop.entity.User;

import com.example.onlineshop.util.PersistenceUtil;

import jakarta.enterprise.context.ApplicationScoped;

import jakarta.inject.Inject;

import jakarta.persistence.EntityManager;

import jakarta.transaction.Transactional;

import java.io.Serializable;

import java.util.List;

@ApplicationScoped

public class PointService implements Serializable {

@Inject

private PersistenceUtil persistenceUtil;

@Transactional

public void createPointRecord(User user, int points, String source) {

EntityManager em = persistenceUtil.getEntityManager();

try {

em.getTransaction().begin();

PointRecord pointRecord = new PointRecord();

pointRecord.setUser(user);

pointRecord.setPoints(points);

pointRecord.setSource(source);

pointRecord.setCreatedAt(java.time.LocalDateTime.now());

em.persist(pointRecord);

user.setPoints(user.getPoints() + points);

em.merge(user);

em.getTransaction().commit();

} catch (Exception e) {

if (em.getTransaction().isActive()) {

em.getTransaction().rollback();

}

throw new RuntimeException("创建积分记录失败: " + e.getMessage());

} finally {

em.close();

}

}

public List<PointRecord> findPointRecordsByUser(User user) {

EntityManager em = persistenceUtil.getEntityManager();

try {

return em.createQuery(

"SELECT pr FROM PointRecord pr WHERE pr.user.id = :userId ORDER BY pr.createdAt DESC",

PointRecord.class)

.setParameter("userId", user.getId())

.getResultList();

} finally {

em.close();

}

}

}

**ProductService.java**

package com.example.onlineshop.service;

import com.example.onlineshop.entity.Product;

import com.example.onlineshop.util.PersistenceUtil;

import jakarta.enterprise.context.ApplicationScoped;

import jakarta.persistence.EntityManager;

import java.util.List;

@ApplicationScoped

public class ProductService {

private EntityManager em;

public ProductService() {

this.em = PersistenceUtil.getEntityManager();

}

// 添加商品

public void addProduct(Product product) {

try {

em.getTransaction().begin();

em.persist(product);

em.getTransaction().commit();

} catch (Exception e) {

if (em.getTransaction().isActive()) {

em.getTransaction().rollback();

}

throw new RuntimeException("Failed to add product", e);

}

}

// 更新商品

public void updateProduct(Product product) {

try {

em.getTransaction().begin();

em.merge(product);

em.getTransaction().commit();

} catch (Exception e) {

if (em.getTransaction().isActive()) {

em.getTransaction().rollback();

}

throw new RuntimeException("Failed to update product", e);

}

}

// 删除商品

public void deleteProduct(Long id) {

try {

em.getTransaction().begin();

Product product = em.find(Product.class, id);

if (product != null) {

em.remove(product);

}

em.getTransaction().commit();

} catch (Exception e) {

if (em.getTransaction().isActive()) {

em.getTransaction().rollback();

}

throw new RuntimeException("Failed to delete product", e);

}

}

// 查询单个商品

public Product findProductById(Long id) {

return em.find(Product.class, id);

}

// 查询所有商品

public List<Product> findAllProducts() {

return em.createQuery("SELECT p FROM Product p", Product.class)

.getResultList();

}

// 按分类查询商品

public List<Product> findProductsByCategory(String category) {

return em.createQuery("SELECT p FROM Product p WHERE p.category = :category", Product.class)

.setParameter("category", category)

.getResultList();

}

}

**UserService.java**

package com.example.onlineshop.service;

import com.example.onlineshop.entity.PointRecord;

import com.example.onlineshop.entity.User;

import com.example.onlineshop.util.PersistenceUtil;

import jakarta.enterprise.context.ApplicationScoped;

import jakarta.inject.Inject;

import jakarta.persistence.EntityManager;

import jakarta.persistence.NoResultException;

import jakarta.transaction.Transactional;

import java.util.List;

@ApplicationScoped

public class UserService {

@Inject

private PersistenceUtil persistenceUtil;

public User authenticate(String username, String password) {

EntityManager em = persistenceUtil.getEntityManager();

try {

return em.createQuery("SELECT u FROM User u WHERE u.username = :username AND u.password = :password", User.class)

.setParameter("username", username)

.setParameter("password", password)

.getSingleResult();

} catch (NoResultException e) {

return null;

} finally {

em.close();

}

}

public User findByUsername(String username) {

EntityManager em = persistenceUtil.getEntityManager();

try {

return em.createQuery("SELECT u FROM User u WHERE u.username = :username", User.class)

.setParameter("username", username)

.getSingleResult();

} catch (NoResultException e) {

return null;

} finally {

em.close();

}

}

public User findById(Long id) {

EntityManager em = persistenceUtil.getEntityManager();

try {

return em.find(User.class, id);

} finally {

em.close();

}

}

public List<Long> findAllUserIds() {

EntityManager em = persistenceUtil.getEntityManager();

try {

return em.createQuery("SELECT u.id FROM User u", Long.class).getResultList();

} finally {

em.close();

}

}

public List<User> findAllUsers() {

EntityManager em = persistenceUtil.getEntityManager();

try {

return em.createQuery("SELECT u FROM User u ORDER BY u.id", User.class)

.getResultList();

} finally {

em.close();

}

}

@Transactional

public void updateUser(User user) {

EntityManager em = persistenceUtil.getEntityManager();

try {

em.getTransaction().begin();

em.merge(user);

em.getTransaction().commit();

} catch (Exception e) {

if (em.getTransaction().isActive()) {

em.getTransaction().rollback();

}

throw new RuntimeException("更新用户失败: " + e.getMessage());

} finally {

em.close();

}

}

public List<PointRecord> findPointRecordsByUser(Long userId) {

EntityManager em = persistenceUtil.getEntityManager();

try {

return em.createQuery("SELECT pr FROM PointRecord pr WHERE pr.user.id = :userId ORDER BY pr.createdAt DESC", PointRecord.class)

.setParameter("userId", userId)

.getResultList();

} finally {

em.close();

}

}

public void registerUser(User user) {

EntityManager em = persistenceUtil.getEntityManager();

try {

em.getTransaction().begin();

em.persist(user);

em.getTransaction().commit();

} finally {

em.close();

}

}

}

1. 工具类

**LocalDateTimeConverter.java**

package com.example.onlineshop.util;

import jakarta.faces.component.UIComponent;

import jakarta.faces.context.FacesContext;

import jakarta.faces.convert.Converter;

import jakarta.faces.convert.ConverterException;

import jakarta.faces.convert.FacesConverter;

import java.time.LocalDateTime;

import java.time.format.DateTimeFormatter;

import java.time.format.DateTimeParseException;

@FacesConverter(value = "localDateTimeConverter")

public class LocalDateTimeConverter implements Converter<LocalDateTime> {

private static final DateTimeFormatter FORMATTER = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");

@Override

public LocalDateTime getAsObject(FacesContext context, UIComponent component, String value) {

if (value == null || value.trim().isEmpty()) {

return null;

}

try {

return LocalDateTime.parse(value, FORMATTER);

} catch (DateTimeParseException e) {

throw new ConverterException("Invalid LocalDateTime format: " + value, e);

}

}

@Override

public String getAsString(FacesContext context, UIComponent component, LocalDateTime value) {

if (value == null) {

return "";

}

try {

return value.format(FORMATTER);

} catch (Exception e) {

throw new ConverterException("Failed to format LocalDateTime: " + value, e);

}

}

}

**PersistenceUtil.java**

package com.example.onlineshop.util;

import jakarta.persistence.EntityManager;

import jakarta.persistence.EntityManagerFactory;

import jakarta.persistence.Persistence;

public class PersistenceUtil {

private static final EntityManagerFactory emf;

static {

try {

emf = Persistence.createEntityManagerFactory("onlineshopPU");

} catch (Exception e) {

throw new RuntimeException("Failed to initialize EntityManagerFactory", e);

}

}

public static EntityManager getEntityManager() {

return emf.createEntityManager();

}

public static void close() {

if (emf != null && emf.isOpen()) {

emf.close();

}

}

}

1. 网页服务类

**PointWebService.java**

package com.example.onlineshop.webservice;

import com.example.onlineshop.entity.PointRecord;

import com.example.onlineshop.entity.User;

import com.example.onlineshop.service.PointService;

import com.example.onlineshop.service.UserService;

import jakarta.inject.Inject;

import jakarta.jws.WebService;

import jakarta.xml.ws.WebServiceException;

import java.util.List;

import java.util.logging.Logger;

@WebService(endpointInterface = "com.example.onlineshop.webservice.PointWebServiceApi")

public class PointWebService implements PointWebServiceApi {

private static final Logger LOGGER = Logger.getLogger(PointWebService.class.getName());

@Inject

private UserService userService;

@Inject

private PointService pointService;

@Override

public int getUserPoints(Long userId) {

LOGGER.info("Fetching points for userId: " + userId);

User user = userService.findById(userId);

if (user == null) {

String errorMsg = "User not found for ID: " + userId;

LOGGER.warning(errorMsg);

throw new WebServiceException(errorMsg);

}

LOGGER.info("User found: " + user.getUsername() + ", Points: " + user.getPoints());

return user.getPoints();

}

@Override

public List<PointRecord> getUserPointRecords(Long userId) {

LOGGER.info("Fetching point records for userId: " + userId);

User user = userService.findById(userId);

if (user == null) {

String errorMsg = "User not found for ID: " + userId;

LOGGER.warning(errorMsg);

throw new WebServiceException(errorMsg);

}

List<PointRecord> records = pointService.findPointRecordsByUser(user);

LOGGER.info("Found " + records.size() + " point records for user: " + user.getUsername());

return records;

}

}

**PointWebServiceApi.java**

package com.example.onlineshop.webservice;

import com.example.onlineshop.entity.PointRecord;

import jakarta.jws.WebMethod;

import jakarta.jws.WebParam;

import jakarta.jws.WebService;

import java.util.List;

@WebService

public interface PointWebServiceApi {

@WebMethod

int getUserPoints(@WebParam(name = "userId") Long userId);

@WebMethod

List<PointRecord> getUserPointRecords(@WebParam(name = "userId") Long userId);

}

beans.xml

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://xmlns.jcp.org/xml/ns/javaee"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/beans\_2\_0.xsd"

bean-discovery-mode="all">

</beans>

persistence.xml

<?xml version="1.0" encoding="UTF-8"?>

<persistence version="3.0" xmlns="https://jakarta.ee/xml/ns/persistence"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="https://jakarta.ee/xml/ns/persistence https://jakarta.ee/xml/ns/persistence/persistence\_3\_0.xsd">

<persistence-unit name="onlineshopPU" transaction-type="RESOURCE\_LOCAL">

<provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>

<class>com.example.onlineshop.entity.User</class>

<class>com.example.onlineshop.entity.Product</class>

<class>com.example.onlineshop.entity.Order</class>

<class>com.example.onlineshop.entity.OrderItem</class>

<class>com.example.onlineshop.entity.PointRecord</class>

<class>com.example.onlineshop.entity.CartItem</class>

<!-- 其他实体 -->

<properties>

<property name="jakarta.persistence.jdbc.url" value="jdbc:mysql://localhost:3306/mydb

?useSSL=false&amp;serverTimezone=UTC"/>

<property name="jakarta.persistence.jdbc.user" value="root"/>

<property name="jakarta.persistence.jdbc.password" value="123456"/>

<property name="jakarta.persistence.jdbc.driver" value="com.mysql.cj.jdbc.Driver"/>

<property name="hibernate.dialect" value="org.hibernate.dialect.MySQLDialect"/>

<property name="hibernate.hbm2ddl.auto" value="update"/>

<property name="hibernate.show\_sql" value="true"/>

</properties>

</persistence-unit>

</persistence>

add\_product.xhtml

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:h="http://xmlns.jcp.org/jsf/html"

xmlns:f="http://xmlns.jcp.org/jsf/core"

xmlns:ui="http://xmlns.jcp.org/jsf/facelets">

<h:head>

<title>#{productController.product.id == null ? '添加商品' : '编辑商品'} - 在线购物平台</title>

<style>

body {

font-family: 'Segoe UI', Arial, sans-serif;

background: linear-gradient(135deg, #6b7280, #3b82f6);

margin: 0;

padding: 2rem;

min-height: 100vh;

color: #1f2937;

}

.container {

background: white;

border-radius: 12px;

box-shadow: 0 8px 24px rgba(0, 0, 0, 0.15);

padding: 2.5rem;

max-width: 600px;

margin: 0 auto;

animation: fadeIn 0.8s ease-out;

opacity: 0;

animation-fill-mode: forwards;

}

@keyframes fadeIn {

from { opacity: 0; transform: translateY(20px); }

to { opacity: 1; transform: translateY(0); }

}

.title {

font-size: 2rem;

color: #1f2937;

text-align: center;

margin-bottom: 1rem;

}

.welcome-text {

color: #4b5563;

text-align: center;

font-size: 1rem;

margin-bottom: 2rem;

}

.form-group {

margin-bottom: 1.8rem;

}

.form-label {

display: block;

color: #374151;

font-weight: 500;

font-size: 0.95rem;

margin-bottom: 0.5rem;

}

.form-input, .form-textarea, .form-select {

width: 100%;

padding: 0.9rem;

border: 2px solid #e5e7eb;

border-radius: 8px;

font-size: 1rem;

background: #f9fafb;

transition: border-color 0.3s ease, box-shadow 0.3s ease;

box-sizing: border-box;

}

.form-textarea {

resize: vertical;

min-height: 100px;

}

.form-input:focus, .form-textarea:focus, .form-select:focus {

border-color: #3b82f6;

background: white;

box-shadow: 0 0 0 3px rgba(59, 130, 246, 0.2);

outline: none;

}

.btn {

padding: 0.9rem 2rem;

background: linear-gradient(to right, #3b82f6, #2563eb);

color: white;

border: none;

border-radius: 8px;

font-size: 1.1rem;

font-weight: 500;

cursor: pointer;

transition: transform 0.2s, box-shadow 0.2s;

width: 100%;

}

.btn:hover {

transform: translateY(-2px);

box-shadow: 0 4px 12px rgba(59, 130, 246, 0.4);

}

.btn:active {

transform: translateY(0);

}

.btn-cancel {

background: #ef4444;

}

.btn-cancel:hover {

background: #dc2626;

box-shadow: 0 4px 12px rgba(239, 68, 68, 0.4);

}

.error-messages {

color: #b91c1c;

background: #fef2f2;

padding: 0.8rem;

border-radius: 6px;

margin-bottom: 1.5rem;

font-size: 0.9rem;

border-left: 4px solid #dc2626;

}

.link {

color: #3b82f6;

text-decoration: none;

font-weight: 500;

transition: color 0.2s;

}

.link:hover {

color: #2563eb;

text-decoration: underline;

}

.link-container {

text-align: center;

margin-top: 1.5rem;

}

.button-group {

display: flex;

gap: 1rem;

}

</style>

</h:head>

<h:body>

<div class="container">

<h1 class="title">#{productController.product.id == null ? '添加商品' : '编辑商品'}</h1>

<p class="welcome-text">为平台添加新商品或更新现有商品信息</p>

<!-- 错误信息 -->

<h:messages globalOnly="false" styleClass="error-messages" layout="table"/>

<h:form>

<div class="form-group">

<h:outputLabel for="name" value="商品名称" styleClass="form-label"/>

<h:inputText id="name" value="#{productController.product.name}"

required="true" requiredMessage="商品名称不能为空"

styleClass="form-input">

<f:validateLength minimum="2" maximum="100"/>

</h:inputText>

</div>

<div class="form-group">

<h:outputLabel for="description" value="描述" styleClass="form-label"/>

<h:inputTextarea id="description" value="#{productController.product.description}"

styleClass="form-textarea"/>

</div>

<div class="form-group">

<h:outputLabel for="price" value="价格" styleClass="form-label"/>

<h:inputText id="price" value="#{productController.product.price}"

required="true" requiredMessage="价格不能为空"

styleClass="form-input">

<f:validateDoubleRange minimum="0.01" maximum="1000000"/>

<f:convertNumber minFractionDigits="2"/>

</h:inputText>

</div>

<div class="form-group">

<h:outputLabel for="stock" value="库存" styleClass="form-label"/>

<h:inputText id="stock" value="#{productController.product.stock}"

required="true" requiredMessage="库存不能为空"

styleClass="form-input">

<f:validateLongRange minimum="0" maximum="100000"/>

</h:inputText>

</div>

<div class="form-group">

<h:outputLabel for="category" value="分类" styleClass="form-label"/>

<h:selectOneMenu id="category" value="#{productController.product.category}"

required="true" requiredMessage="分类不能为空"

styleClass="form-select">

<f:selectItem itemLabel="电子产品" itemValue="电子产品"/>

<f:selectItem itemLabel="服装" itemValue="服装"/>

<f:selectItem itemLabel="食品" itemValue="食品"/>

</h:selectOneMenu>

</div>

<div class="form-group">

<h:outputLabel for="imageUrl" value="图片 URL" styleClass="form-label"/>

<h:inputText id="imageUrl" value="#{productController.product.imageUrl}"

styleClass="form-input"/>

</div>

<div class="form-group button-group">

<h:commandButton value="保存" action="#{productController.addProduct}" styleClass="btn"/>

<h:commandButton value="取消" action="#{productController.cancelAddProduct}"

styleClass="btn btn-cancel" immediate="true"/>

</div>

</h:form>

<div class="link-container">

<h:link outcome="product\_list" styleClass="link">

返回商品列表

</h:link>

</div>

</div>

</h:body>

</html>

cart.xhtml

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:h="http://xmlns.jcp.org/jsf/html"

xmlns:f="http://xmlns.jcp.org/jsf/core"

xmlns:ui="http://xmlns.jcp.org/jsf/facelets">

<h:head>

<title>购物车 - 在线购物平台</title>

<style>

body {

font-family: 'Segoe UI', Arial, sans-serif;

background: linear-gradient(135deg, #6b7280, #3b82f6);

margin: 0;

padding: 2rem;

min-height: 100vh;

color: #1f2937;

}

.container {

background: white;

border-radius: 12px;

box-shadow: 0 8px 24px rgba(0, 0, 0, 0.15);

padding: 2.5rem;

max-width: 1000px;

margin: 0 auto;

animation: fadeIn 0.8s ease-out;

opacity: 0;

animation-fill-mode: forwards;

}

@keyframes fadeIn {

from { opacity: 0; transform: translateY(20px); }

to { opacity: 1; transform: translateY(0); }

}

.title {

font-size: 2rem;

color: #1f2937;

text-align: center;

margin-bottom: 1rem;

}

.cart-table {

width: 100%;

border-collapse: collapse;

margin-bottom: 2rem;

}

.cart-table th, .cart-table td {

padding: 1rem;

text-align: left;

border-bottom: 1px solid #e5e7eb;

}

.cart-table th {

background: #f3f4f6;

color: #1f2937;

font-weight: 600;

}

.cart-table td {

color: #4b5563;

}

.cart-table img {

width: 80px;

height: 80px;

object-fit: cover;

border-radius: 8px;

}

.quantity-input {

width: 60px;

padding: 0.5rem;

border: 1px solid #d1d5db;

border-radius: 6px;

text-align: center;

}

.action-btn {

padding: 0.5rem 1rem;

border-radius: 6px;

font-size: 0.9rem;

text-decoration: none;

text-align: center;

transition: background-color 0.2s;

cursor: pointer;

}

.action-btn.update {

background: #3b82f6;

color: white;

}

.action-btn.update:hover {

background: #2563eb;

}

.action-btn.remove {

background: #ef4444;

color: white;

}

.action-btn.remove:hover {

background: #dc2626;

}

.action-btn.submit {

background: #10b981;

color: white;

padding: 0.75rem 1.5rem;

font-size: 1rem;

}

.action-btn.submit:hover {

background: #059669;

}

.total-section {

text-align: right;

margin-top: 1.5rem;

color: #1f2937;

font-size: 1.1rem;

}

.points-section {

margin: 1.5rem 0;

padding: 1rem;

background: #f9fafb;

border-radius: 8px;

}

.points-section label {

font-weight: 500;

color: #1f2937;

margin-right: 1rem;

}

.points-section input {

padding: 0.5rem;

border: 1px solid #d1d5db;

border-radius: 6px;

width: 100px;

}

.points-section p {

color: #4b5563;

margin: 0.5rem 0;

}

.empty-message {

text-align: center;

color: #4b5563;

font-size: 1.1rem;

margin: 2rem 0;

}

.link {

color: #3b82f6;

text-decoration: none;

font-weight: 500;

transition: color 0.2s;

}

.link:hover {

color: #2563eb;

text-decoration: underline;

}

.error-messages {

color: #b91c1c;

background: #fef2f2;

padding: 0.8rem;

border-radius: 6px;

margin-bottom: 1.5rem;

font-size: 0.9rem;

border-left: 4px solid #dc2626;

}

</style>

</h:head>

<h:body>

<div class="container">

<h1 class="title">我的购物车</h1>

<!-- 错误信息 -->

<h:messages globalOnly="false" styleClass="error-messages" layout="table"/>

<h:form>

<h:dataTable value="#{orderController.cartItems}" var="item" styleClass="cart-table"

rendered="#{not empty orderController.cartItems}">

<h:column>

<f:facet name="header">商品</f:facet>

<h:graphicImage value="#{item.product.imageUrl}" style="width: 80px; height: 80px;"/>

<h:outputText value="#{item.product.name}"/>

</h:column>

<h:column>

<f:facet name="header">单价</f:facet>

<h:outputText value="￥#{item.product.price}"/>

</h:column>

<h:column>

<f:facet name="header">数量</f:facet>

<h:inputText value="#{item.quantity}" styleClass="quantity-input"/>

<h:commandLink value="更新" action="#{orderController.updateCartItem(item.product.id, item.quantity)}"

styleClass="action-btn update"/>

</h:column>

<h:column>

<f:facet name="header">小计</f:facet>

<h:outputText value="￥#{item.quantity \* item.product.price}"/>

</h:column>

<h:column>

<f:facet name="header">操作</f:facet>

<h:commandLink value="删除" action="#{orderController.removeFromCart(item.product.id)}"

styleClass="action-btn remove"/>

</h:column>

</h:dataTable>

<!-- 空状态 -->

<h:panelGroup rendered="#{empty orderController.cartItems}">

<p class="empty-message">

购物车为空，

<h:commandLink value="去购物吧！" action="product\_list.faces" styleClass="link">

<f:param name="faces-redirect" value="true"/>

</h:commandLink>

</p>

</h:panelGroup>

<!-- 积分抵扣 -->

<h:panelGroup rendered="#{not empty orderController.cartItems and authController.isLoggedIn()}">

<div class="points-section">

<p>当前可用积分：#{authController.currentUser.points}</p>

<h:outputLabel value="使用积分（100积分 = 1元）：" for="pointsToUse"/>

<h:inputText id="pointsToUse" value="#{orderController.pointsToUse}" styleClass="points-section input"/>

<h:commandButton value="应用积分" action="#{orderController.applyPoints}" styleClass="action-btn update"/>

<h:panelGroup rendered="#{orderController.pointsDiscount > 0}">

<p>积分抵扣：￥#{orderController.pointsDiscount}</p>

</h:panelGroup>

</div>

</h:panelGroup>

<!-- 总计和提交 -->

<h:panelGroup rendered="#{not empty orderController.cartItems}">

<div class="total-section">

<p>总计：￥#{orderController.cartTotal}</p>

<h:commandButton value="提交订单" action="#{orderController.submitOrder}"

styleClass="action-btn submit"/>

</div>

</h:panelGroup>

</h:form>

<h:form>

<h:commandLink value="返回商品列表" action="#{orderController.goToProductList}" styleClass="link"/>

</h:form>

</div>

</h:body>

</html>

edit\_product.xhtml

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:h="http://xmlns.jcp.org/jsf/html"

xmlns:f="http://xmlns.jcp.org/jsf/core">

<h:head>

<title>编辑商品 - 在线购物平台</title>

<style>

body {

font-family: 'Segoe UI', Arial, sans-serif;

background: linear-gradient(135deg, #6b7280, #3b82f6);

margin: 0;

padding: 0;

display: flex;

justify-content: center;

align-items: center;

min-height: 100vh;

overflow: hidden;

}

.login-container {

background: white;

border-radius: 12px;

box-shadow: 0 8px 24px rgba(0, 0, 0, 0.15);

padding: 2.5rem;

width: 100%;

max-width: 420px;

animation: fadeIn 0.8s ease-out;

opacity: 0;

animation-fill-mode: forwards;

}

@keyframes fadeIn {

from { opacity: 0; transform: translateY(20px); }

to { opacity: 1; transform: translateY(0); }

}

.login-title {

color: #1f2937;

text-align: center;

font-size: 1.8rem;

margin-bottom: 0.5rem;

}

.welcome-text {

color: #4b5563;

text-align: center;

font-size: 1rem;

margin-bottom: 2rem;

}

.form-group {

margin-bottom: 1.8rem;

}

.form-label {

display: block;

color: #374151;

font-weight: 500;

font-size: 0.95rem;

margin-bottom: 0.5rem;

}

.form-input, .form-textarea, .form-select {

width: 100%;

padding: 0.9rem;

border: 2px solid #e5e7eb;

border-radius: 8px;

font-size: 1rem;

background: #f9fafb;

transition: border-color 0.3s ease, box-shadow 0.3s ease;

box-sizing: border-box;

}

.form-textarea {

height: 100px;

resize: vertical;

}

.form-input:focus, .form-textarea:focus, .form-select:focus {

border-color: #3b82f6;

background: white;

box-shadow: 0 0 0 3px rgba(59, 130, 246, 0.2);

outline: none;

}

.btn {

width: 100%;

padding: 0.9rem;

background: linear-gradient(to right, #3b82f6, #2563eb);

color: white;

border: none;

border-radius: 8px;

font-size: 1.1rem;

font-weight: 500;

cursor: pointer;

transition: transform 0.2s, box-shadow 0.2s;

}

.btn:hover {

transform: translateY(-2px);

box-shadow: 0 4px 12px rgba(59, 130, 246, 0.4);

}

.btn:active {

transform: translateY(0);

}

.link-container {

text-align: center;

margin-top: 1.5rem;

}

.link {

color: #3b82f6;

text-decoration: none;

font-weight: 500;

transition: color 0.2s;

}

.link:hover {

color: #2563eb;

text-decoration: underline;

}

.error-messages {

color: #b91c1c;

background: #fef2f2;

padding: 0.8rem;

border-radius: 6px;

margin-bottom: 1.5rem;

font-size: 0.9rem;

border-left: 4px solid #dc2626;

}

/\* 输入框抖动动画（验证失败时触发） \*/

.shake {

animation: shake 0.4s cubic-bezier(0.36, 0.07, 0.19, 0.97) both;

}

@keyframes shake {

10%, 90% { transform: translateX(-1px); }

20%, 80% { transform: translateX(2px); }

30%, 50%, 70% { transform: translateX(-4px); }

40%, 60% { transform: translateX(4px); }

}

</style>

<h:outputScript>

/\* <![CDATA[ \*/

document.addEventListener('DOMContentLoaded', function() {

var messages = document.querySelector('.error-messages');

if (messages && messages.innerHTML.trim() !== '') {

var inputs = document.querySelectorAll('.form-input, .form-textarea, .form-select');

inputs.forEach(function(input) {

input.classList.add('shake');

setTimeout(function() { input.classList.remove('shake'); }, 400);

});

}

});

/\* ]]> \*/

</h:outputScript>

</h:head>

<h:body>

<div class="login-container">

<h1 class="login-title">编辑商品</h1>

<p class="welcome-text">更新商品信息以保持最新</p>

<!-- 错误信息 -->

<h:messages globalOnly="false" styleClass="error-messages" layout="table"/>

<h:form>

<div class="form-group">

<h:outputLabel for="name" value="名称" styleClass="form-label"/>

<h:inputText id="name" value="#{productController.product.name}"

required="true" requiredMessage="名称不能为空"

styleClass="form-input"/>

</div>

<div class="form-group">

<h:outputLabel for="description" value="描述" styleClass="form-label"/>

<h:inputTextarea id="description" value="#{productController.product.description}"

styleClass="form-textarea"/>

</div>

<div class="form-group">

<h:outputLabel for="price" value="价格" styleClass="form-label"/>

<h:inputText id="price" value="#{productController.product.price}"

required="true" requiredMessage="价格不能为空"

styleClass="form-input">

<f:convertNumber minFractionDigits="2"/>

</h:inputText>

</div>

<div class="form-group">

<h:outputLabel for="stock" value="库存" styleClass="form-label"/>

<h:inputText id="stock" value="#{productController.product.stock}"

required="true" requiredMessage="库存不能为空"

styleClass="form-input"/>

</div>

<div class="form-group">

<h:outputLabel for="category" value="分类" styleClass="form-label"/>

<h:selectOneMenu id="category" value="#{productController.product.category}"

required="true" requiredMessage="分类不能为空"

styleClass="form-select">

<f:selectItem itemLabel="电子产品" itemValue="电子产品"/>

<f:selectItem itemLabel="服装" itemValue="服装"/>

<f:selectItem itemLabel="食品" itemValue="食品"/>

</h:selectOneMenu>

</div>

<div class="form-group">

<h:outputLabel for="imageUrl" value="图片 URL" styleClass="form-label"/>

<h:inputText id="imageUrl" value="#{productController.product.imageUrl}"

styleClass="form-input"/>

</div>

<div class="form-group">

<h:commandButton value="保存" action="#{productController.saveEdit}"

styleClass="btn"/>

</div>

<div class="link-container">

<h:commandLink value="返回列表" action="product\_list.faces"

styleClass="link">

<f:param name="faces-redirect" value="true"/>

</h:commandLink>

</div>

</h:form>

</div>

</h:body>

</html>

login.xhtml

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:h="http://xmlns.jcp.org/jsf/html"

xmlns:f="http://xmlns.jcp.org/jsf/core"

xmlns:ui="http://xmlns.jcp.org/jsf/facelets">

<h:head>

<title>用户登录 - 在线购物平台</title>

<style>

body {

font-family: 'Segoe UI', Arial, sans-serif;

background: linear-gradient(135deg, #6b7280, #3b82f6);

margin: 0;

padding: 0;

display: flex;

justify-content: center;

align-items: center;

min-height: 100vh;

overflow: hidden;

}

.login-container {

background: white;

border-radius: 12px;

box-shadow: 0 8px 24px rgba(0, 0, 0, 0.15);

padding: 2.5rem;

width: 100%;

max-width: 420px;

animation: fadeIn 0.8s ease-out;

opacity: 0;

animation-fill-mode: forwards;

}

@keyframes fadeIn {

from { opacity: 0; transform: translateY(20px); }

to { opacity: 1; transform: translateY(0); }

}

.login-title {

color: #1f2937;

text-align: center;

font-size: 1.8rem;

margin-bottom: 0.5rem;

}

.welcome-text {

color: #4b5563;

text-align: center;

font-size: 1rem;

margin-bottom: 2rem;

}

.form-group {

margin-bottom: 1.8rem;

}

.form-label {

display: block;

color: #374151;

font-weight: 500;

font-size: 0.95rem;

margin-bottom: 0.5rem;

}

.form-input {

width: 100%;

padding: 0.9rem;

border: 2px solid #e5e7eb;

border-radius: 8px;

font-size: 1rem;

background: #f9fafb;

transition: border-color 0.3s ease, box-shadow 0.3s ease;

box-sizing: border-box;

}

.form-input:focus {

border-color: #3b82f6;

background: white;

box-shadow: 0 0 0 3px rgba(59, 130, 246, 0.2);

outline: none;

}

.btn {

width: 100%;

padding: 0.9rem;

background: linear-gradient(to right, #3b82f6, #2563eb);

color: white;

border: none;

border-radius: 8px;

font-size: 1.1rem;

font-weight: 500;

cursor: pointer;

transition: transform 0.2s, box-shadow 0.2s;

}

.btn:hover {

transform: translateY(-2px);

box-shadow: 0 4px 12px rgba(59, 130, 246, 0.4);

}

.btn:active {

transform: translateY(0);

}

.link-container {

text-align: center;

margin-top: 1.5rem;

}

.link {

color: #3b82f6;

text-decoration: none;

font-weight: 500;

transition: color 0.2s;

}

.link:hover {

color: #2563eb;

text-decoration: underline;

}

.error-messages {

color: #b91c1c;

background: #fef2f2;

padding: 0.8rem;

border-radius: 6px;

margin-bottom: 1.5rem;

font-size: 0.9rem;

border-left: 4px solid #dc2626;

}

.shake {

animation: shake 0.4s cubic-bezier(0.36, 0.07, 0.19, 0.97) both;

}

@keyframes shake {

10%, 90% { transform: translateX(-1px); }

20%, 80% { transform: translateX(2px); }

30%, 50%, 70% { transform: translateX(-4px); }

40%, 60% { transform: translateX(4px); }

}

</style>

<h:outputScript>

/\* <![CDATA[ \*/

document.addEventListener('DOMContentLoaded', function() {

var messages = document.querySelector('.error-messages');

if (messages && messages.innerHTML.trim() !== '') {

var inputs = document.querySelectorAll('.form-input');

inputs.forEach(function(input) {

input.classList.add('shake');

setTimeout(function() { input.classList.remove('shake'); }, 400);

});

}

});

/\* ]]> \*/

</h:outputScript>

</h:head>

<h:body>

<div class="login-container">

<h1 class="login-title">欢迎登录</h1>

<p class="welcome-text">加入我们的在线购物平台，享受便捷购物体验！</p>

<!-- 显示错误信息 -->

<h:messages globalOnly="false" styleClass="error-messages" layout="table"/>

<h:form>

<div class="form-group">

<h:outputLabel for="username" value="用户名" styleClass="form-label"/>

<h:inputText id="username" value="#{orderController.username}"

required="true" requiredMessage="用户名不能为空"

styleClass="form-input"/>

</div>

<div class="form-group">

<h:outputLabel for="password" value="密码" styleClass="form-label"/>

<h:inputSecret id="password" value="#{orderController.password}"

required="true" requiredMessage="密码不能为空"

styleClass="form-input"/>

</div>

<div class="form-group">

<h:commandButton value="立即登录" action="#{orderController.login}"

styleClass="btn"/>

</div>

<div class="link-container">

<h:commandLink value="没有账号？立即注册" action="register.faces"

styleClass="link">

<f:param name="faces-redirect" value="true"/>

</h:commandLink>

</div>

</h:form>

</div>

</h:body>

</html>

order\_list.xhtml

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:h="http://xmlns.jcp.org/jsf/html"

xmlns:f="http://xmlns.jcp.org/jsf/core"

xmlns:ui="http://xmlns.jcp.org/jsf/facelets">

<h:head>

<title>我的订单 - 在线购物平台</title>

<style>

body {

font-family: 'Segoe UI', Arial, sans-serif;

background: linear-gradient(135deg, #6b7280, #3b82f6);

margin: 0;

padding: 2rem;

min-height: 100vh;

color: #1f2937;

}

.container {

background: white;

border-radius: 12px;

box-shadow: 0 8px 24px rgba(0, 0, 0, 0.15);

padding: 2.5rem;

max-width: 1200px;

margin: 0 auto;

animation: fadeIn 0.8s ease-out;

opacity: 0;

animation-fill-mode: forwards;

}

@keyframes fadeIn {

from { opacity: 0; transform: translateY(20px); }

to { opacity: 1; transform: translateY(0); }

}

.title {

font-size: 2rem;

color: #1f2937;

text-align: center;

margin-bottom: 1rem;

}

.welcome-text {

color: #4b5563;

text-align: center;

font-size: 1rem;

margin-bottom: 2rem;

}

.order-grid {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(280px, 1fr));

gap: 1.5rem;

margin-bottom: 2rem;

}

.order-card {

background: #f9fafb;

border-radius: 8px;

padding: 1.5rem;

box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);

transition: transform 0.3s ease, box-shadow 0.3s ease;

}

.order-card:hover {

transform: translateY(-5px);

box-shadow: 0 4px 16px rgba(0, 0, 0, 0.15);

}

.order-card h3 {

margin: 0 0 0.5rem;

font-size: 1.2rem;

color: #1f2937;

}

.order-card p {

margin: 0.3rem 0;

color: #4b5563;

font-size: 0.95rem;

}

.order-actions {

display: flex;

gap: 0.5rem;

margin-top: 1rem;

}

.action-btn {

padding: 0.5rem 1rem;

border-radius: 6px;

font-size: 0.9rem;

text-decoration: none;

text-align: center;

transition: background-color 0.2s;

}

.action-btn.pay {

background: #10b981;

color: white;

}

.action-btn.pay:hover {

background: #059669;

}

.action-btn.cancel {

background: #ef4444;

color: white;

}

.action-btn.cancel:hover {

background: #dc2626;

}

.action-btn.points {

background: #3b82f6;

color: white;

}

.action-btn.points:hover {

background: #2563eb;

}

.empty-message {

text-align: center;

color: #4b5563;

font-size: 1.1rem;

margin: 2rem 0;

}

.link {

color: #3b82f6;

text-decoration fournir

font-weight: 500;

transition: color 0.2s;

}

.link:hover {

color: #2563eb;

text-decoration: underline;

}

.link-container {

text-align: center;

margin-top: 1.5rem;

}

.error-messages {

color: #b91c1c;

background: #fef2f2;

padding: 0.8rem;

border-radius: 6px;

margin-bottom: 1.5rem;

font-size: 0.9rem;

border-left: 4px solid #dc2626;

}

</style>

</h:head>

<h:body>

<div class="container">

<h1 class="title">我的订单</h1>

<p class="welcome-text">查看和管理你的订单记录</p>

<!-- 错误信息 -->

<h:messages globalOnly="false" styleClass="error-messages" layout="table"/>

<h:form>

<!-- 订单卡片 -->

<div class="order-grid">

<ui:repeat value="#{orderController.orders}" var="order">

<div class="order-card">

<h3>订单 #{order.id}</h3>

<p>日期：<h:outputText value="#{order.orderDate}">

<f:converter converterId="localDateTimeConverter"/>

</h:outputText></p>

<p>金额：￥#{order.totalAmount}</p>

<p>状态：#{order.status}</p>

<p>使用积分：#{order.pointsUsed > 0 ? order.pointsUsed : '无'}</p>

<p>商品：

<ui:repeat value="#{order.orderItems}" var="item">

#{item.product.name} (x#{item.quantity})<br/>

</ui:repeat>

</p>

<div class="order-actions">

<h:commandLink value="支付" action="#{orderController.payOrder(order.id)}"

styleClass="action-btn pay"

rendered="#{order.status == 'PENDING'}"/>

<h:commandLink value="取消" action="#{orderController.cancelOrder(order.id)}"

styleClass="action-btn cancel"

rendered="#{order.status == 'PENDING'}"/>

</div>

</div>

</ui:repeat>

</div>

<!-- 空状态 -->

<h:panelGroup rendered="#{orderController.orders == null or empty orderController.orders}">

<p class="empty-message">

暂无订单，

<h:commandLink value="去购物吧！" action="product\_list.faces" styleClass="link">

<f:param name="faces-redirect" value="true"/>

</h:commandLink>

</p>

</h:panelGroup>

</h:form>

<h:form>

<div class="link-container">

<h:commandLink value="返回商品列表" action="#{orderController.goToProductList}" styleClass="link"/>

<h:commandLink value="我的积分" action="#{pointController.goToPoints}" styleClass="link"

rendered="#{authController.isLoggedIn()}"/>

</div>

</h:form>

</div>

</h:body>

</html>

points.xhtml

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:h="http://xmlns.jcp.org/jsf/html"

xmlns:f="http://xmlns.jcp.org/jsf/core"

xmlns:ui="http://xmlns.jcp.org/jsf/facelets">

<h:head>

<title>我的积分 - 在线购物平台</title>

<style>

body {

font-family: 'Segoe UI', Arial, sans-serif;

background: linear-gradient(135deg, #6b7280, #3b82f6);

margin: 0;

padding: 2rem;

min-height: 100vh;

color: #1f2937;

}

.container {

background: white;

border-radius: 12px;

box-shadow: 0 8px 24px rgba(0, 0, 0, 0.15);

padding: 2.5rem;

max-width: 800px;

margin: 0 auto;

animation: fadeIn 0.8s ease-out;

opacity: 0;

animation-fill-mode: forwards;

}

@keyframes fadeIn {

from { opacity: 0; transform: translateY(20px); }

to { opacity: 1; transform: translateY(0); }

}

.title {

font-size: 2rem;

color: #1f2937;

text-align: center;

margin-bottom: 1rem;

}

.points-balance {

text-align: center;

font-size: 1.5rem;

color: #10b981;

margin-bottom: 2rem;

}

.points-table {

width: 100%;

border-collapse: collapse;

margin-bottom: 2rem;

}

.points-table th, .points-table td {

padding: 1rem;

text-align: left;

border-bottom: 1px solid #e5e7eb;

}

.points-table th {

background: #f3f4f6;

color: #1f2937;

font-weight: 600;

}

.points-table td {

color: #4b5563;

}

.empty-message {

text-align: center;

color: #4b5563;

font-size: 1.1rem;

margin: 2rem 0;

}

.link {

color: #3b82f6;

text-decoration: none;

font-weight: 500;

transition: color 0.2s;

}

.link:hover {

color: #2563eb;

text-decoration: underline;

}

.link-container {

text-align: center;

margin-top: 1.5rem;

}

.error-messages {

color: #b91c1c;

background: #fef2f2;

padding: 0.8rem;

border-radius: 6px;

margin-bottom: 1.5rem;

font-size: 0.9rem;

border-left: 4px solid #dc2626;

}

</style>

</h:head>

<h:body>

<div class="container">

<h1 class="title">我的积分</h1>

<p class="points-balance">当前积分：#{authController.currentUser.points}</p>

<!-- 错误信息 -->

<h:messages globalOnly="false" styleClass="error-messages" layout="table"/>

<h:form>

<h:dataTable value="#{pointController.pointRecords}" var="record" styleClass="points-table"

rendered="#{not empty pointController.pointRecords}">

<h:column>

<f:facet name="header">时间</f:facet>

<h:outputText value="#{record.createdAt}">

<f:converter converterId="localDateTimeConverter"/>

</h:outputText>

</h:column>

<h:column>

<f:facet name="header">积分</f:facet>

<h:outputText value="#{record.points}"/>

</h:column>

<h:column>

<f:facet name="header">来源</f:facet>

<h:outputText value="#{record.source}"/>

</h:column>

</h:dataTable>

<!-- 空状态 -->

<h:panelGroup rendered="#{empty pointController.pointRecords}">

<p class="empty-message">暂无积分记录</p>

</h:panelGroup>

<div class="link-container">

<h:commandLink value="返回商品列表" action="#{orderController.goToProductList}" styleClass="link"/>

<h:commandLink value="我的订单" action="#{orderController.goToOrders}" styleClass="link"/>

</div>

</h:form>

</div>

</h:body>

</html>

product\_list.xhtml

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:h="http://xmlns.jcp.org/jsf/html"

xmlns:f="http://xmlns.jcp.org/jsf/core"

xmlns:ui="http://xmlns.jcp.org/jsf/facelets">

<h:head>

<title>商品列表 - 在线购物平台</title>

<style>

body {

font-family: 'Segoe UI', Arial, sans-serif;

background: linear-gradient(135deg, #6b7280, #3b82f6);

margin: 0;

padding: 2rem;

min-height: 100vh;

color: #1f2937;

}

.container {

background: white;

border-radius: 12px;

box-shadow: 0 8px 24px rgba(0, 0, 0, 0.15);

padding: 2.5rem;

max-width: 1200px;

margin: 0 auto;

animation: fadeIn 0.8s ease-out;

opacity: 0;

animation-fill-mode: forwards;

}

@keyframes fadeIn {

from { opacity: 0; transform: translateY(20px); }

to { opacity: 1; transform: translateY(0); }

}

.title {

font-size: 2rem;

color: #1f2937;

text-align: center;

margin-bottom: 1rem;

}

.welcome-text {

color: #4b5563;

text-align: center;

font-size: 1rem;

margin-bottom: 2rem;

}

.user-info {

display: flex;

justify-content: space-between;

align-items: center;

margin-bottom: 2rem;

font-size: 1rem;

color: #4b5563;

}

.user-links a {

color: #3b82f6;

text-decoration: none;

margin-left: 1rem;

font-weight: 500;

transition: color 0.2s;

}

.user-links a:hover {

color: #2563eb;

text-decoration: underline;

}

.filter-form {

margin-bottom: 2rem;

display: flex;

align-items: center;

gap: 1rem;

}

.form-label {

color: #374151;

font-weight: 500;

font-size: 0.95rem;

}

.form-select {

padding: 0.75rem;

border: 2px solid #e5e7eb;

border-radius: 8px;

font-size: 1rem;

background: #f9fafb;

transition: border-color 0.3s ease, box-shadow 0.3s ease;

}

.form-select:focus {

border-color: #3b82f6;

box-shadow: 0 0 0 3px rgba(59, 130, 246, 0.2);

outline: none;

}

.btn {

padding: 0.75rem 1.5rem;

background: linear-gradient(to right, #3b82f6, #2563eb);

color: white;

border: none;

border-radius: 8px;

font-size: 1rem;

font-weight: 500;

cursor: pointer;

transition: transform 0.2s, box-shadow 0.2s;

}

.btn:hover {

transform: translateY(-2px);

box-shadow: 0 4px 12px rgba(59, 130, 246, 0.4);

}

.btn:active {

transform: translateY(0);

}

.product-grid {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(280px, 1fr));

gap: 1.5rem;

margin-bottom: 2rem;

}

.product-card {

background: #f9fafb;

border-radius: 8px;

padding: 1.5rem;

box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);

transition: transform 0.3s ease, box-shadow 0.3s ease;

}

.product-card:hover {

transform: translateY(-5px);

box-shadow: 0 4px 16px rgba(0, 0, 0, 0.15);

}

.product-card h3 {

margin: 0 0 0.5rem;

font-size: 1.2rem;

color: #1f2937;

}

.product-card p {

margin: 0.3rem 0;

color: #4b5563;

font-size: 0.95rem;

}

.product-image {

width: 100%;

height: 200px;

object-fit: cover;

border-radius: 6px;

margin-bottom: 1rem;

display: block;

}

.product-image-placeholder {

width: 100%;

height: 200px;

background: #e5e7eb;

border-radius: 6px;

margin-bottom: 1rem;

display: flex;

align-items: center;

justify-content: center;

color: #6b7280;

font-size: 0.9rem;

}

.product-actions {

display: flex;

gap: 0.5rem;

margin-top: 1rem;

}

.action-btn {

padding: 0.5rem 1rem;

border-radius: 6px;

font-size: 0.9rem;

text-decoration: none;

text-align: center;

transition: background-color 0.2s;

}

.action-btn.cart {

background: #3b82f6;

color: white;

}

.action-btn.cart:hover {

background: #2563eb;

}

.action-btn.edit {

background: #10b981;

color: white;

}

.action-btn.edit:hover {

background: #059669;

}

.action-btn.delete {

background: #ef4444;

color: white;

}

.action-btn.delete:hover {

background: #dc2626;

}

.add-product {

text-align: center;

}

.add-product a {

color: #3b82f6;

text-decoration: none;

font-weight: 500;

font-size: 1rem;

transition: color 0.2s;

}

.add-product a:hover {

color: #2563eb;

text-decoration: underline;

}

.error-messages {

color: #b91c1c;

background: #fef2f2;

padding: 0.8rem;

border-radius: 6px;

margin-bottom: 1.5rem;

font-size: 0.9rem;

border-left: 4px solid #dc2626;

}

</style>

</h:head>

<h:body>

<div class="container">

<h1 class="title">商品列表</h1>

<p class="welcome-text">发现适合你的优质商品！</p>

<!-- 错误信息 -->

<h:messages globalOnly="false" styleClass="error-messages" layout="table"/>

<!-- 用户信息 -->

<h:form>

<div class="user-info">

<span>欢迎，#{authController.isLoggedIn() ? authController.currentUser.nickname : '游客'}！</span>

<div class="user-links">

<h:commandLink value="购物车" action="#{orderController.goToCart}"

styleClass="link" rendered="#{authController.isLoggedIn()}"/>

<h:commandLink value="我的订单" action="#{orderController.goToOrders}"

styleClass="link" rendered="#{authController.isLoggedIn()}"/>

<h:commandLink value="我的积分" action="#{pointController.goToPoints}"

styleClass="link" rendered="#{authController.isLoggedIn()}"/>

<h:commandLink value="管理用户" action="user\_management?faces-redirect=true"

styleClass="link" rendered="#{authController.isLoggedIn() and authController.currentUser.role == 'ADMIN'}"/>

<h:commandLink value="退出登录" action="#{authController.logout}"

styleClass="link" rendered="#{authController.isLoggedIn()}"/>

<h:commandLink value="登录" action="login.faces"

styleClass="link" rendered="#{not authController.isLoggedIn()}">

<f:param name="faces-redirect" value="true"/>

</h:commandLink>

</div>

</div>

</h:form>

<!-- 分类筛选 -->

<h:form styleClass="filter-form">

<label class="form-label">分类：</label>

<h:selectOneMenu value="#{productController.selectedCategory}" styleClass="form-select">

<f:selectItem itemLabel="全部" itemValue=""/>

<f:selectItem itemLabel="电子产品" itemValue="电子产品"/>

<f:selectItem itemLabel="服装" itemValue="服装"/>

<f:selectItem itemLabel="食品" itemValue="食品"/>

</h:selectOneMenu>

<h:commandButton value="筛选" styleClass="btn"/>

</h:form>

<!-- 商品卡片 -->

<h:form>

<div class="product-grid">

<ui:repeat value="#{productController.productsByCategory}" var="prod">

<div class="product-card">

<h:graphicImage value="#{prod.imageUrl != null and prod.imageUrl != '' ? prod.imageUrl : '/resources/images/placeholder.jpg'}"

styleClass="product-image"

alt="#{prod.name}"/>

<h3>#{prod.name}</h3>

<p>价格：￥#{prod.price}</p>

<p>库存：#{prod.stock}</p>

<p>分类：#{prod.category}</p>

<div class="product-actions">

<h:commandButton value="加入购物车" action="#{orderController.addToCart(prod.id, 1)}"

styleClass="action-btn cart" rendered="#{authController.isLoggedIn()}"/>

<h:commandLink value="编辑" action="#{productController.editProduct(prod.id)}"

styleClass="action-btn edit"

rendered="#{authController.isLoggedIn() and authController.currentUser.role == 'ADMIN'}"/>

<h:commandLink value="删除" action="#{productController.deleteProduct(prod.id)}"

styleClass="action-btn delete"

rendered="#{authController.isLoggedIn() and authController.currentUser.role == 'ADMIN'}"/>

</div>

</div>

</ui:repeat>

</div>

</h:form>

<!-- 添加商品 -->

<h:form>

<div class="add-product">

<h:commandLink value="添加新商品" action="#{productController.goToAddProduct}"

styleClass="link"

rendered="#{authController.isLoggedIn() and authController.currentUser.role == 'ADMIN'}"/>

</div>

</h:form>

</div>

</h:body>

</html>

register.xhtml

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:h="http://xmlns.jcp.org/jsf/html"

xmlns:f="http://xmlns.jcp.org/jsf/core"

xmlns:ui="http://xmlns.jcp.org/jsf/facelets">

<h:head>

<title>用户注册 - 在线购物平台</title>

<style>

body {

font-family: 'Segoe UI', Arial, sans-serif;

background: linear-gradient(135deg, #6b7280, #3b82f6);

margin: 0;

padding: 2rem;

min-height: 100vh;

color: #1f2937;

}

.container {

background: white;

border-radius: 12px;

box-shadow: 0 8px 24px rgba(0, 0, 0, 0.15);

padding: 2.5rem;

max-width: 600px;

margin: 0 auto;

animation: fadeIn 0.8s ease-out;

opacity: 0;

animation-fill-mode: forwards;

}

@keyframes fadeIn {

from { opacity: 0; transform: translateY(20px); }

to { opacity: 1; transform: translateY(0); }

}

.title {

font-size: 2rem;

color: #1f2937;

text-align: center;

margin-bottom: 1rem;

}

.welcome-text {

color: #4b5563;

text-align: center;

font-size: 1rem;

margin-bottom: 2rem;

}

.form-group {

margin-bottom: 1.8rem;

}

.form-label {

display: block;

color: #374151;

font-weight: 500;

font-size: 0.95rem;

margin-bottom: 0.5rem;

}

.form-input {

width: 100%;

padding: 0.9rem;

border: 2px solid #e5e7eb;

border-radius: 8px;

font-size: 1rem;

background: #f9fafb;

transition: border-color 0.3s ease, box-shadow 0.3s ease;

box-sizing: border-box;

}

.form-input:focus {

border-color: #3b82f6;

background: white;

box-shadow: 0 0 0 3px rgba(59, 130, 246, 0.2);

outline: none;

}

.btn {

padding: 0.9rem 2rem;

background: linear-gradient(to right, #3b82f6, #2563eb);

color: white;

border: none;

border-radius: 8px;

font-size: 1.1rem;

font-weight: 500;

cursor: pointer;

transition: transform 0.2s, box-shadow 0.2s;

width: 100%;

}

.btn:hover {

transform: translateY(-2px);

box-shadow: 0 4px 12px rgba(59, 130, 246, 0.4);

}

.btn:active {

transform: translateY(0);

}

.error-messages {

color: #b91c1c;

background: #fef2f2;

padding: 0.8rem;

border-radius: 6px;

margin-bottom: 1.5rem;

font-size: 0.9rem;

border-left: 4px solid #dc2626;

}

.link {

color: #3b82f6;

text-decoration: none;

font-weight: 500;

transition: color 0.2s;

cursor: pointer;

}

.link:hover {

color: #2563eb;

text-decoration: underline;

}

.link-container {

text-align: center;

margin-top: 1.5rem;

}

</style>

</h:head>

<h:body>

<div class="container">

<h1 class="title">用户注册</h1>

<p class="welcome-text">创建一个账号，开启你的购物之旅！</p>

<!-- 错误信息 -->

<h:messages globalOnly="false" styleClass="error-messages" layout="table"/>

<h:form>

<div class="form-group">

<h:outputLabel for="username" value="用户名" styleClass="form-label"/>

<h:inputText id="username" value="#{userController.user.username}"

required="true" requiredMessage="用户名不能为空"

styleClass="form-input"/>

</div>

<div class="form-group">

<h:outputLabel for="password" value="密码" styleClass="form-label"/>

<h:inputSecret id="password" value="#{userController.user.password}"

required="true" requiredMessage="密码不能为空"

styleClass="form-input"/>

</div>

<div class="form-group">

<h:outputLabel for="confirmPassword" value="确认密码" styleClass="form-label"/>

<h:inputSecret id="confirmPassword" value="#{userController.confirmPassword}"

required="true" requiredMessage="确认密码不能为空"

styleClass="form-input"/>

</div>

<div class="form-group">

<h:outputLabel for="nickname" value="昵称" styleClass="form-label"/>

<h:inputText id="nickname" value="#{userController.user.nickname}"

required="true" requiredMessage="昵称不能为空"

styleClass="form-input"/>

</div>

<div class="form-group">

<h:commandButton value="注册" action="#{userController.register}" styleClass="btn"/>

</div>

</h:form>

<h:form>

<div class="link-container">

<h:commandLink value="已有账号？立即登录" action="login" styleClass="link">

<f:param name="faces-redirect" value="true"/>

</h:commandLink>

</div>

</h:form>

</div>

</h:body>

</html>

user\_management.xhtml

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:h="http://xmlns.jcp.org/jsf/html"

xmlns:f="http://xmlns.jcp.org/jsf/core"

xmlns:ui="http://xmlns.jcp.org/jsf/facelets">

<h:head>

<title>用户管理 - 管理在线购物平台</title>

<style>

body {

font-family: 'Segoe UI', Arial, sans-serif;

background: linear-gradient(135deg, #6b7280, #3b82f6);

margin: 0;

padding: 2rem;

min-height: 100vh;

}

.container {

background: white;

border-radius: 12px;

box-shadow: 0 8px 24px rgba(0, 0, 0, 0.15);

padding: 2.5rem;

max-width: 1200px;

margin: 0 auto;

animation: fadeIn 0.8s ease-out;

}

@keyframes fadeIn {

from { opacity: 0; transform: translateY(20px); }

to { opacity: 1; transform: translateY(0); }

}

.title {

color: #1f2937;

font-size: 1.8rem;

margin-bottom: 1rem;

}

.form-group {

margin-bottom: 1.8rem;

}

.form-label {

display: block;

color: #374151;

font-weight: 500;

font-size: 0.95rem;

margin-bottom: 0.5rem;

}

.form-input {

width: 100%;

padding: 0.9rem;

border: 2px solid #e5e7eb;

border-radius: 8px;

font-size: 1rem;

background: #f9fafb;

transition: border-color 0.3s ease, box-shadow 0.3s ease;

box-sizing: border-box;

}

.form-input:focus {

border-color: #3b82f6;

background: white;

box-shadow: 0 0 0 3px rgba(59, 130, 246, 0.2);

outline: none;

}

.btn {

display: inline-block;

padding: 0.9rem 1.5rem;

background: linear-gradient(to right, #3b82f6, #2563eb);

color: white;

border: none;

border-radius: 8px;

font-size: 1rem;

font-weight: 500;

cursor: pointer;

text-decoration: none;

transition: transform 0.2s, box-shadow 0.2s;

margin: 0.5rem;

}

.btn:hover {

transform: translateY(-2px);

box-shadow: 0 4px 12px rgba(59, 130, 246, 0.4);

}

.btn:active {

transform: translateY(0);

}

table {

width: 100%;

border-collapse: collapse;

margin: 1rem 0;

}

th, td {

padding: 1rem;

text-align: left;

border-bottom: 1px solid #e5e7eb;

}

th {

background: #f9fafb;

font-weight: 600;

}

.error-messages {

color: #b91c1c;

background: #fef2f2;

padding: 0.8rem;

border-radius: 6px;

margin-bottom: 1.5rem;

font-size: 0.9rem;

border-left: 4px solid #dc2626;

}

.shake {

animation: shake 0.4s cubic-bezier(0.36, 0.07, 0.19, 0.97) both;

}

@keyframes shake {

10%, 90% { transform: translateX(-1px); }

20%, 80% { transform: translateX(2px); }

30%, 50%, 70% { transform: translateX(-4px); }

40%, 60% { transform: translateX(4px); }

}

</style>

<h:outputScript>

/\* <![CDATA[ \*/

document.addEventListener('DOMContentLoaded', function() {

var messages = document.querySelector('.error-messages');

if (messages && messages.innerHTML.trim() !== '') {

var inputs = document.querySelectorAll('.form-input');

inputs.forEach(function(input) {

input.classList.add('shake');

setTimeout(function() { input.classList.remove('shake'); }, 400);

});

}

});

/\* ]]> \*/

</h:outputScript>

</h:head>

<h:body>

<div class="container">

<h:form rendered="#{authController.loggedIn and authController.currentUser.role == 'ADMIN'}">

<h1 class="title">用户管理</h1>

<h:messages globalOnly="false" styleClass="error-messages"/>

<!-- 用户列表 -->

<h2>所有用户</h2>

<h:dataTable value="#{userController.users}" var="user">

<h:column>

<f:facet name="header">ID</f:facet>

#{user.id}

</h:column>

<h:column>

<f:facet name="header">用户名</f:facet>

#{user.username}

</h:column>

<h:column>

<f:facet name="header">昵称</f:facet>

#{user.nickname}

</h:column>

<h:column>

<f:facet name="header">积分</f:facet>

#{user.points}

</h:column>

<h:column>

<f:facet name="header">角色</f:facet>

#{user.role}

</h:column>

<h:column>

<f:facet name="header">操作</f:facet>

<h:commandLink value="编辑" action="#{userController.editUser(user.id)}" styleClass="btn"/>

</h:column>

</h:dataTable>

<!-- 编辑用户 -->

<h:panelGroup rendered="#{userController.selectedUser != null}">

<h2>编辑用户</h2>

<div class="form-group">

<h:outputLabel for="nickname" value="昵称" styleClass="form-label"/>

<h:inputText id="nickname" value="#{userController.selectedUser.nickname}"

required="true" requiredMessage="昵称不能为空"

styleClass="form-input"/>

</div>

<div class="form-group">

<h:outputLabel for="points" value="积分" styleClass="form-label"/>

<h:inputText id="points" value="#{userController.selectedUser.points}"

required="true" requiredMessage="积分不能为空"

styleClass="form-input">

<f:convertNumber integerOnly="true"/>

<f:validateLongRange minimum="0"/>

</h:inputText>

</div>

<div class="form-group">

<h:outputLabel for="role" value="角色" styleClass="form-label"/>

<h:selectOneMenu id="role" value="#{userController.selectedUser.role}"

required="true" requiredMessage="角色不能为空"

styleClass="form-input">

<f:selectItem itemValue="ADMIN" itemLabel="管理员"/>

<f:selectItem itemValue="USER" itemLabel="用户"/>

</h:selectOneMenu>

</div>

<h:commandButton value="保存" action="#{userController.saveUser}" styleClass="btn"/>

<h:commandLink value="取消" action="#{userController.cancelEdit}" styleClass="btn"/>

</h:panelGroup>

<!-- 返回 -->

<h:commandLink value="返回商品列表" action="product\_list?faces-redirect=true" styleClass="btn"/>

<h:commandLink value="退出登录" action="#{authController.logout}" styleClass="btn"/>

</h:form>

<h:panelGroup rendered="#{!authController.loggedIn}">

<h1>请登录</h1>

<h:commandLink value="返回登录" action="login?faces-redirect=true" styleClass="btn"/>

</h:panelGroup>

<h:panelGroup rendered="#{authController.loggedIn and authController.currentUser.role != 'ADMIN'}">

<h1>权限不足</h1>

<h:outputText value="仅管理员可访问此页面"/>

<h:commandLink value="退出登录" action="#{authController.logout}" styleClass="btn"/>

</h:panelGroup>

</div>

</h:body>

</html>

faces-config.xml

<?xml version="1.0" encoding="UTF-8"?>

<faces-config xmlns="http://xmlns.jcp.org/xml/ns/javaee"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-facesconfig\_2\_3.xsd"

version="2.3">

<navigation-rule>

<from-view-id>/pages/register.xhtml</from-view-id>

<navigation-case>

<from-outcome>login</from-outcome>

<to-view-id>/pages/login.xhtml</to-view-id>

<redirect/>

</navigation-case>

</navigation-rule>

<navigation-rule>

<from-view-id>/pages/login.xhtml</from-view-id>

<navigation-case>

<from-outcome>product\_list</from-outcome>

<to-view-id>/pages/product\_list.xhtml</to-view-id>

<redirect/>

</navigation-case>

</navigation-rule>

<navigation-rule>

<from-view-id>/pages/product\_list.xhtml</from-view-id>

<navigation-case>

<from-outcome>user\_management</from-outcome>

<to-view-id>/pages/user\_management.xhtml</to-view-id>

<redirect/>

</navigation-case>

<navigation-case>

<from-outcome>points</from-outcome>

<to-view-id>/pages/points.xhtml</to-view-id>

<redirect/>

</navigation-case>

</navigation-rule>

<navigation-rule>

<from-view-id>\*</from-view-id>

<navigation-case>

<from-outcome>product\_list</from-outcome>

<to-view-id>/pages/product\_list.xhtml</to-view-id>

<redirect/>

</navigation-case>

<navigation-case>

<from-outcome>product\_list\_user</from-outcome>

<to-view-id>/pages/product\_list\_user.xhtml</to-view-id>

<redirect/>

</navigation-case>

<navigation-case>

<from-outcome>user\_management</from-outcome>

<to-view-id>/pages/user\_management.xhtml</to-view-id>

<redirect/>

</navigation-case>

</navigation-rule>

</faces-config>

sun-jaxws.xml

<?xml version="1.0" encoding="UTF-8"?>

<endpoints xmlns="http://java.sun.com/xml/ns/jax-ws/ri/runtime" version="2.0">

<endpoint name="PointWebService"

implementation="com.example.onlineshop.webservice.PointWebService"

url-pattern="/ws/points"/>

</endpoints>

web.xml

<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns="https://jakarta.ee/xml/ns/jakartaee"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee https://jakarta.ee/xml/ns/jakartaee/web-app\_5\_0.xsd"

version="5.0">

<!-- JSF 配置 -->

<servlet>

<servlet-name>Faces Servlet</servlet-name>

<servlet-class>jakarta.faces.webapp.FacesServlet</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>Faces Servlet</servlet-name>

<url-pattern>\*.xhtml</url-pattern>

</servlet-mapping>

<!-- JAX-WS 配置 -->

<listener>

<listener-class>com.sun.xml.ws.transport.http.servlet.WSServletContextListener</listener-class>

</listener>

<servlet>

<servlet-name>PointWebService</servlet-name>

<servlet-class>com.sun.xml.ws.transport.http.servlet.WSServlet</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>PointWebService</servlet-name>

<url-pattern>/ws/points</url-pattern>

</servlet-mapping>

<!-- 欢迎页面 -->

<welcome-file-list>

<welcome-file>pages/login.xhtml</welcome-file>

</welcome-file-list>

</web-app>